1971

An algorithm for the solution of concave-convex games

Gunn, Lee Fredric; Tocha, Peter

Monterey, California. Naval Postgraduate School

http://hdl.handle.net/10945/15562

AN ALGORITHM FOR THE SOLUTION OF CONCAVE-CONVEX GAMES

LEE FREDRIC GUNN

AND

PETER TOCHA

United States
Naval Postgraduate School

THESIS

AN ALGORITHM FOR THE SOLUTION
OF CONCAVE-CONVEX GAMES

by

Lee Fredric Gunn

and

Peter Tocha

Thesis Advisor: J. M. Danskin, Jr.

September 1971

An Algorithm for the Solution

of Concave-Convex Games

by

Lee Fredric Gunn
Lieutenant, United States Navy
B.A., University of California, Los Angeles, 1965

and

Peter Tocha
Kapitänleutnant, German Navy

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL

ABSTRACT

A master's thesis which discusses the solution of concave-convex games. An algorithm is developed, a computer program written and applied to an anti-submarine warfare force allocation problem as an illustration. Techniques for handling concave-convex problems in high dimensions are included.

TABLE OF CONTENTS

I. INTRODUCTION -- 5

A. A GAME THEORY APPROACH TO RESOURCE ALLOCATIONS -------------------------------- 5

B. CONTENT OF THE PAPER, BY CHAPTER ----------- 6

II. MATHEMATICAL CONCEPTS ------------------------- 7

A. GENERAL ----------------------------------- 7

B. THE ALGORITHM FOR THE SOLUTION OF CONCAVE-CONVEX GAMES -------------------------------- 8

1. The Derivative Game -------------------- 8

2. The Lemma of the Alternative ----------- 9

a. The Brown-Robinson Iterative Process in the "Auxiliary Game" ------------ 9

b. Statement of the Lemma ------------- 11

3. The Corollary of the Alternative ------- 12

4. The Algorithm -------------------------- 13

III. PROGRAMMING ASPECTS ON AN ILLUSTRATIVE EXAMPLE - 18

A. FORMULATION OF THE PROBLEM ----------------- 18

1. The Space X ---------------------------- 20

2. The Space Y ---------------------------- 20

3. The Function F(x,y) -------------------- 21

B. BASIC COMPUTATIONS ------------------------- 22

1. Computation of θ_{him} -------------------- 22

2. Partial Derivatives with Respect to x_{hi}- 23

3. The Value of F(x,y) -------------------- 23

4. Partial Derivatives with Respect to y_{jk}- 23

5. Finding Directions of Increase (Decrease) -------------------------- 23

C. PROGRAMMING NOTES ---------------------------- 24

1. Presentation of Input - The Mission Description ---------------------------- 24

2. The Contraction with Respect to V_{him} --- 27

3. Working in Subspaces ------------------- 27

4. Machine Accuracy Problems -------------- 29

5. Testing the Program -------------------- 30

6. General Comments ---------------------- 31

a. Initial Allocations ---------------- 31

b. Distance Policy -------------------- 32

c. Upper and Lower Bounds ------------- 32

d. Modification of the Objective Function -------------------------- 32

e. Assigning Values V_{him} -------------- 33

f. Choosing ρ -------------------------- 33

IV. SUMMARY ------------------------------------- 35

V. SUGGESTIONS FOR FURTHER STUDY ------------------ 36

APPENDIX: PROGRAMMING FLOWCHART --------------------- 38

COMPUTER PROGRAM LISTING ---------------------------- 51

BIBLIOGRAPHY -- 74

INITIAL DISTRIBUTION LIST --------------------------- 76

FORM DD 1473 -- 77

I. INTRODUCTION

A. A GAME THEORY APPROACH TO RESOURCE ALLOCATIONS

Problems in the allocation of resources can be divided into two descriptive categories: Single-agent problems and adversary problems. Single-agent problems have only one participant optimizing without intelligent opposition. In the solution of adversary problems, opponents work at cross purposes. Each choice of an allocation of resources by one participant must be made in light of those of his opponent(s). Of course, games are adversary problems.

The great interest in game theory as a technique of modelling which followed von Neumann's statement of the fundamental concepts in 1927 [18] continues today. The fascination of the game as a model for conflicts of almost any sort is enhanced by the fact that the solution to a game is entirely independent of assumptions regarding the *actual* behavior of the antagonist(s).

Matrix games and differential games are extensively treated in the literature; some references are noted here. Matrix games or games over the square are discussed in basic form by Williams [21] and a thorough study is done by Karlin [8]. For a first essay in the subject of differential games, see Isaacs [6]. Taylor [16,17] gives additional examples of the modelling of combat operations including search using differential games.

The development of the derivative game is due to Danskin [4]. A very large class of concave-convex problems

yield to solution when the algorithm based on this game is applied. It is with the derivative game and the algorithm evolved from it that this paper is primarily concerned.

B. THE CONTENT OF THE PAPER, BY CHAPTER

Chapter II surveys some of the more important mathematical ideas necessary to the development of the algorithm for the solution of concave-convex games.

Chapter III is concerned with the programming of an anti-submarine warfare force allocation example. The example serves to illustrate both the use of the algorithm and the theory on which it is founded. The problem is formulated and the basic computational steps toward the solution are enumerated. If any fears stem from the formidable appearance of the matrices used in the example, they hopefully will be dispelled by the description of the form for input in the programming notes. Technical matters regarding programming are considered in some detail.

Chapter IV draws together this presentation with some concluding remarks.

Chapter V contains suggestions for further study.

The Appendix consists of the programming flowchart. The Computer Program Listing is included as well.

II. MATHEMATICAL CONCEPTS

A. GENERAL

The fundamental theorem of the theory of games in the form appropriate here is the following:

Suppose $F(x,y)$ is a continuous function on $X \times Y$, where X and Y are compact and convex. Suppose that the set of points $X(y)$ yielding the maximum to F for fixed y is convex for each such y, and that the set of points $Y(x)$ yielding the minimum to F for fixed x is convex for each such x.

Then there exist pure strategy solutions x° and y° satisfying

$$\begin{aligned} F(x^\circ,y) &\geq F(x^\circ,y^\circ), \ \forall\, y \epsilon Y \\ F(x,y^\circ) &\leq F(x^\circ,y^\circ), \ \forall\, x \epsilon X. \end{aligned} \tag{1}$$

A complete proof of the theorem in this form can be found in [7]. Assuming that the conditions for the existence of a solution satisfying (1) are met, the problem can be stated in the form

$$\underset{x}{\text{Max}}\ \underset{y}{\text{Min}}\ F(x,y),$$

which is equivalent to

$$\underset{x}{\text{Max}}\ \phi(x)$$

where $$\phi(x) \equiv \underset{y}{\text{Min}}\ F(x,y).$$

The theorem applies to two-person zero-sum games. Thus,

$$\underset{x}{\text{Max}}\ \underset{y}{\text{Min}}\ F(x,y) = \underset{y}{\text{Min}}\ \underset{x}{\text{Max}}\ F(x,y).$$

One important difficulty arises from the fact that, although F may be smooth, $\phi(x)$ is not in general differentiable in the ordinary sense. Danskin, in [3], has shown that under general conditions on X and Y, there exists a directional derivative in every direction. It is this fact that has provided the key to the solution of problems of the type described.

B. THE ALGORITHM FOR THE SOLUTION OF CONCAVE-CONVEX GAMES

The algorithm to solve games concave in the maximizing player and convex in the minimizing player is developed and presented in great detail in [4]. The following gives a brief survey of those results which are most important for the design of the algorithm; to fill the apparent gaps, a thorough reading of [4] remains necessary.

1. The Derivative Game

Suppose $x^\circ \varepsilon X$. Associate with x° a non-empty set of *admissible directions* γ, $\Gamma(x^\circ)$. $\hat{W}$ is defined as the convex hull (e.g., see [19]) of the set of points

$$W(y) = \{F_{x_1}(x^\circ,y), \cdots, F_{y_k}(x^\circ,y)\},$$

where $y \varepsilon Y(x^\circ)$.

The derivative game then is

$$H(\gamma,\hat{w}) = \gamma \cdot \hat{w}$$

defined over $\Gamma(x^\circ) \: x \: \hat{W}$. The maximizing player maximizes H by choice of $\gamma \varepsilon \Gamma(x^\circ)$, the minimizing player minimizes H by choice of $\hat{w} \varepsilon \hat{W}$.

THEOREM I

A necessary and sufficient condition for the existence of a direction of increase for $\phi(x)$ is that the value of the derivative game defined by H be positive at x°. The γ° which yields the value of the derivative game is a pure strategy.

If the value is positive one can find and use this direction. If the value is non-positive, a direction of increase does not exist, i.e., the solution has been reached.

The application of the derivative game in practice is greatly complicated by the necessity for approximations.

2. The Lemma of the Alternative

The Lemma takes into exact account the approximations involved in the application of the derivative game. It states that a certain process (to be explained below) must either yield a sufficient increase to $\phi(x)$ at a point x° or determine that the point x° is nearly optimal. Before the lemma can be formulated in mathematical terms, some further difficulties and the tools with which to overcome them have to be outlined.

a. The Brown-Robinson Iterative Process in the "Auxiliary Game"

The Brown-Robinson (B-R) process employs the following idea: Let G be the pay-off function. At stage N=0 both players choose arbitrary strategies x° and y°. At stage N=1 the maximizer chooses x^{1} such that G is maximized against y°; then the minimizer chooses y^{1} to minimize G

against x^1, and so forth. At stage N the maximizer chooses x^N as if the minimizer's strategy were an evenly weighted mixture of strategies $y^\circ, \cdots, y^{N-1}$; the minimizer chooses y^N as if the maximizer's strategy were an evenly weighted mixture of strategies $x^\circ, \cdots, x^N$.

For matrix games, Julia Robinson [12] proved that

$$\limsup_{N\to\infty}\left[\frac{1}{N}\sum_{n=0}^{N-1} G(x,y^n) - \frac{1}{N}\sum_{n=0}^{N} G(x^n,y)\right] = 0.$$

Danskin [1] has generalized the proof to hold for two-person zero-sum games with continuous pay-off defined over $X \times Y$, X and Y arbitrary compact spaces. It should be noted here that the B-R process is very slow in convergence when applied directly to finding an approximation to the value of the game defined by (1). However, it is not applied to the basic game in this algorithm but rather to an "auxiliary game" for which an accurate solution is not required.

The derivative game mentioned above cannot be solved directly because the set $Y(x^\circ)$ is not known. All one has is a single element $y \varepsilon Y$ which approximately minimizes $F(x^\circ, y)$. The place of the derivative game, therefore, is taken by the "auxiliary game" employing a modified version of the B-R process described below. This process makes it possible to keep track of the approximations involved and their consequences. The "auxiliary game" is defined as follows:

For any $\varepsilon \geq 0$, denote by $Y_\varepsilon(x)$ the set of $y \varepsilon Y$ such that $F(x,y) = \phi(x) + \varepsilon$. Let $Y_\varepsilon(\Xi) \equiv \bigcup_{x \varepsilon \Xi} Y_\varepsilon(x)$, $\gamma \varepsilon \Gamma(x^\circ)$, $y \varepsilon Y(\Xi)$.

Then $H(\gamma,y) \equiv \dfrac{F(x^\circ + d_o \gamma, y) - \phi(x^\circ)}{d_o}$, for $d_o > 0$, a minimum step size, is a game over $\Gamma(x^\circ) \times Y_\varepsilon(\Xi)$ with γ the maximizing and y the minimizing player. This game has optimal mixed strategies for both players. Applying the idea of approximate optimization to the convergence proof in [1] leads to

THEOREM II

Let γ^N be chosen such that $\frac{1}{N} \sum_{n=0}^{N-1} H(\gamma, y^n)$ is maximized to accuracy ζ, and y^N be chosen such that $\frac{1}{N} \sum_{n=0}^{N} H(\gamma^n, y)$ is minimized to accuracy η.

Then

$$\limsup_{N\to\infty} \left[\frac{1}{N} \sum_{n=0}^{N-1} H(\gamma, y^n) - \frac{1}{N} \sum_{n=0}^{N} H(\gamma^n, y) \right] \leq 2(\zeta + \eta).$$

This holds for continuous H.

Let θ be the maximum oscillation of $\nabla F(x,y)$ over a distance d_o. Then

$$\frac{F(x^\circ + d_o\ \gamma^N, y^n) - F(x^\circ, y^n)}{d_o} \geq \gamma^N\ F(x^\circ, y^n) - \theta$$

The lemma of the alternative now can be formulated.

b. Statement of the Lemma

Suppose $0 < \alpha < \beta$, $y^\circ \varepsilon Y_\varepsilon(x^\circ)$.

Then the generalized B-R process will, at some stage N, determine that one of the two following statements is true:

1. The maximum over $\Gamma(x^\circ)$ of the directional derivative does not exceed β.

2. The point $\bar{x}^N \equiv x^\circ + d_o \bar{\gamma}^N$, where

$$\bar{\gamma}^N = \frac{1}{N} \sum_{n=1}^{N} \gamma^n,$$

and the point $y^N \varepsilon Y_\varepsilon(\bar{x}^N)$, where y^N minimizes $F(\bar{x}^N, y)$ to accuracy ε, satisfy

$$\frac{F(\bar{x}^N, y^N) - F(x^\circ, y^\circ)}{d_o} \geq \alpha - 5\theta - \frac{3}{d_o}.$$

3. <u>The Corollary of the Alternative</u>

Suppose that $F(x,y)$ is concave in x and convex in y. Then the modified B-R process applied at x° will, at some stage N, determine that one of the two following statements is true:

1. The pair $\bar{x} = x^\circ$, $\bar{y} = \bar{y}^N$, where

$$\bar{y}^N = \frac{1}{N+1} \sum_{n=0}^{N} y^n,$$

are approximate optimal strategies for the game defined by F.

2. The point $\bar{x}^N \varepsilon X$ yields an increase to $\phi(x)$ by at least a specified amount.

For details and proof see [4], pp. 36 ff.

Reference [4] continues with a detailed discussion of delicate problems which can only be listed here: The choice of the minimal step size d_o; the problem of accessibility of a point x from a point x°; the problem of obstruction; the choice of α, β, ε, their interaction with each other, and the choice of ρ where ρ is the accuracy to which Max $\phi(x)$ is to approximate the value of the game defined by

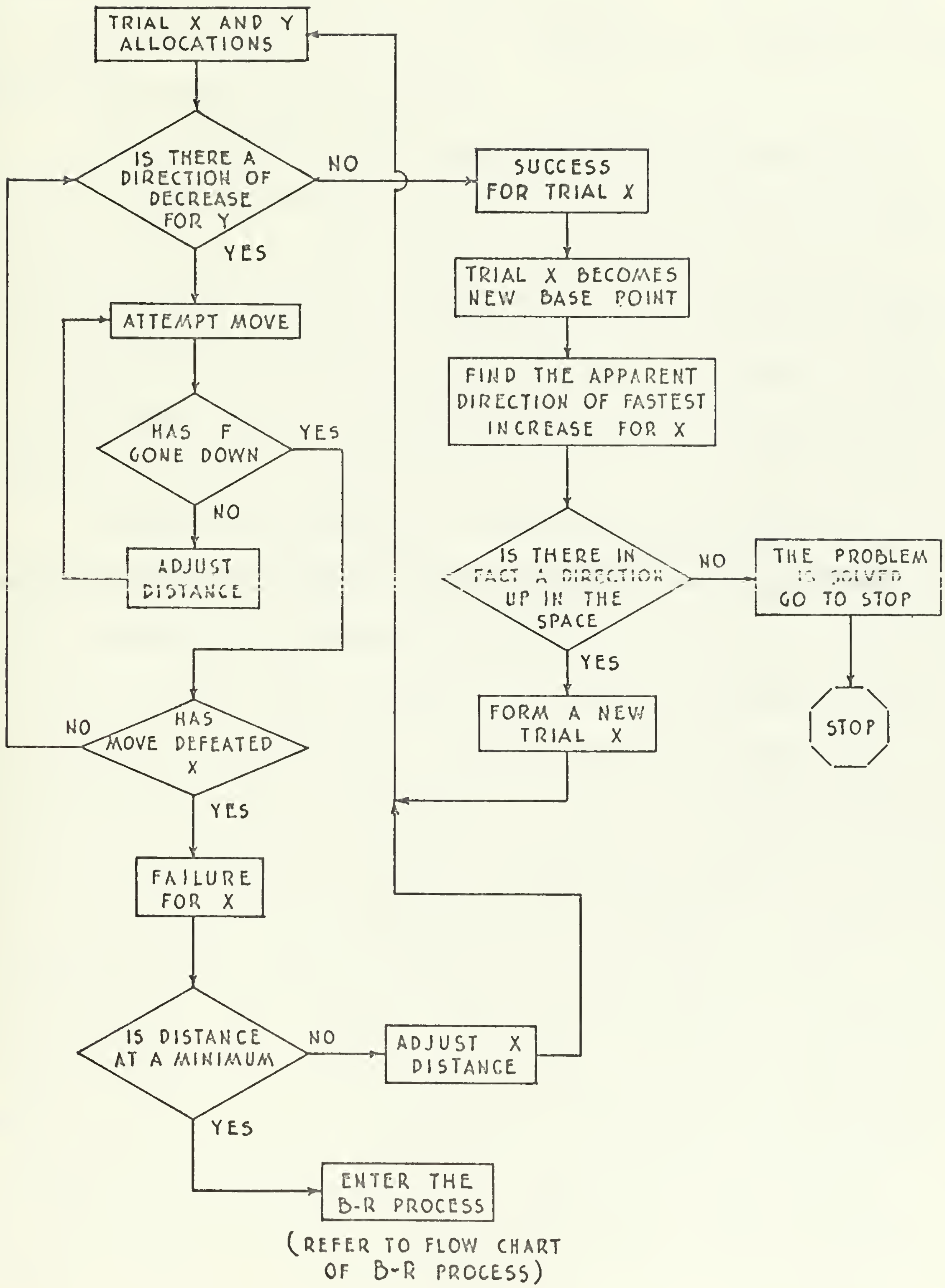
(1). It must be noted that the conditions derived for the selection of these parameters are sufficient.

4. <u>The Algorithm</u>

The algorithm as a consequence of the foregoing mathematical considerations is presented in section 10 and 11 of [4] and will not be reproduced here in detail. A verbal description of its basic structure - depicted in Figure 1 -, however, may be useful:

The maximizing player, called Max, having arrived at a point xx, has a direction of maximal increase γ, obtained either from the derivative game $D_\gamma \phi(xx)$ or from the B-R process in the auxiliary game, and a distance $d \geq d_o$. The minimizing player, called Min, is at a point yy. F(xx,yy) is known. Max makes a proposal to move to a point $x = xx + d\gamma$. Of course, F(x,yy) > F(xx,yy). Min accepts Max's proposal and starts minimizing against x, looking for a direction of maximal decrease g. If there is none, yy is a minimum against x as well as against xx in which case Max will move to the point x. If there is a direction of decrease Min forms a point y = yy + Dg such that $F(x,y) \leq F(x,yy)$. A test is performed to determine whether Min has already "beaten" Max: if $F(xx,yy) \geq F(x,y)$ Min stops the minimization process, and Max discards his proposal x because moving to x will not increase $\phi(x)$. Max halves the distance d and, with the same γ, forms a new trial point x. If F(xx,yy) < F(x,y) Min continues to minimize until either $F(xx,yy) \geq F(x,y)$ or Min can no longer find a direction of decrease. If now

Figure 1. Flowchart of Algorithm.

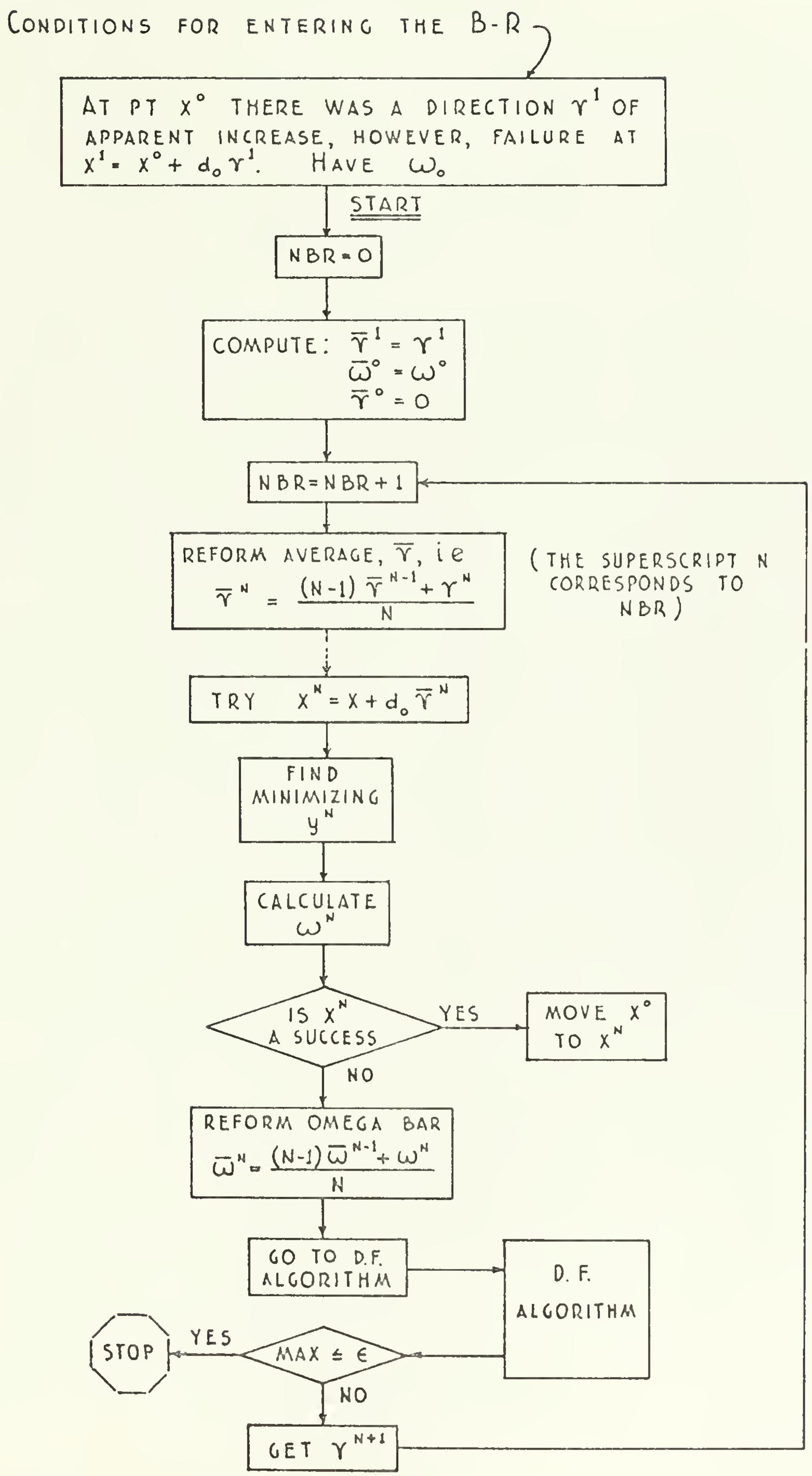


$F(xx,yy) < F(x,y)$, indicating that Min, even after a complete minimization, was not able to "beat" Max, Max moves to the proposed point x realizing a gain for $\phi(x)$. Max then looks for a new direction of increase. The process terminates when such a direction does not exist.

The situation that leads into the Brown-Robinson process is the following one: The proposed points $x = xx+d\gamma$ have been "beaten" by Min until d gets cut down to d_o, in spite of the fact that γ, obtained from the derivative game, is an apparently good direction. If the trial point $x = xx + d_o\gamma$ does not result in a move for Max, the B-R process, Figure 2, is used.

Denote the present γ - the one that so far has lead to a failure for Max by $\bar{\gamma}^1$, and the associated ∇F_x by $\bar{\omega}^\circ$. Minimize F completely against $x = xx + d_o\gamma$. The resulting y then leads to a new ∇F_x which is averaged with the previous $\bar{\omega}^\circ$, giving $\bar{\omega}^1$. Suppose that $\bar{\omega}^1$ as input to the derivative game $D_\gamma\phi(x)$ produces a new γ° such that the value of the derivative game is positive as required. (If such a γ° does not exist the problem is solved). This γ, averaged with $\bar{\gamma}^1$, gives $\bar{\gamma}^2$ which in turn creates a new trial point $\bar{x}^2 = xx + d_o\ \bar{\gamma}^2$. $\bar{x}^2$, or $\bar{x}^N$ in general, then is exposed to Min's reaction as described previously. Once Max finds a direction and an associated trial point that cannot be "beaten" by Min, Max moves and leaves the B-R routine. The y-strategies are also averaged and saved although their average is never used during the computation. In case the

Figure 2. The Brown-Robinson Process.



game defined by $F(x,y)$ is terminated while in the B-R process, this average of the y-strategies represents the optimal solution for the minimizing player.

III. PROGRAMMING ASPECTS OF AN ILLUSTRATIVE EXAMPLE

A. FORMULATION OF THE PROBLEM

An algorithm for the solution of a wide class of concave-convex games over polyhedra has been presented in abbreviated form above. In this chapter that algorithm is applied to a particular game, an anti-submarine warfare force allocation problem.

For this problem, five indices are employed: h, i, j, k, and m. In a meaningful example, the maximum numbers corresponding to these indices might be, respectively: 5, 25, 10, 500, and 5. The indices have the following meanings:

h: Submarine type

i: Submarine mission

j: Type of antisubmarine weapon (or vehicle)

k: Place in which submarine and weapon encounter one another

m: Stage of the submarine mission.

An additional index is used. $\ell(k)$ is the "kind of place." A "kind of place" might be defined by a particular set of weapon employment parameters. These include the tactical and the natural environment. The natural environment consists of oceanographic and meteorological conditions. Examples of the tactical environment are destroyers in an ASW screen and patrol aircraft in barrier patrol. The "kind of place" in which an encounter occurs impacts on the outcome

of an encounter between submarine and weapon. A reasonable number of "kinds of places" in the present context might be 25.

A submarine mission is described by a matrix $||E_{hijkm}||$. This matrix has as its elements real numbers denoting the extent to which a submarine of type h at stage m of mission i is exposed to a weapon of type j at the k^{th} place. The effects of these weapons and thus of the encounters are characterized by a "technical" matrix $||C_{hj\ell(k)}||$ in the following meaning: $\exp[-C_{hj\ell(k)}\ y_{jk}]$ is the probability that a submarine of type h survives one exposure to y units of weapons of type j at a "kind of place" $\ell(k)$. These encounters are assumed to be mutually independent. Note that the probability of survival to the m^{th} mission stage is conditioned on the completion of the previous stages. Suppose that there are y_{jk} units of force of type j at the k^{th} place. Then the probability of a submarine's completing stage m of mission i is

$$\prod_{m' \leq m} \exp[-E_{hijkm'}\ C_{hj\ell(k)}\ y_{jk}],$$

which, due to independence of the events, equals $\exp\,[-\theta_{him}]$, where

$$\theta_{him} = \sum_{j} \sum_{\substack{k \\ m' \leq m}} E_{hijkm'}\ C_{hj\ell(k)}\ y_{jk}.$$

Now, by carrying out a premultiplication,

$$A_{hijkm} = E_{hijkm}\ C_{hj\ell(k)},$$

the exponent becomes

$$\theta_{him} = \sum_{j} \sum_{\substack{k \\ m' \leq m}} A_{hijkm'} \, y_{jk}.$$

In this example, the vast majority of the E_{hijkm}, and therefore of the A_{hijkm}, are zero. If 3000 non-zero A_{hijkm} are allowed, each type of submarine can be employed on ten different missions and undergo up to 60 encounters with antisubmarine weapon systems.

1. The Space X

Let x_{hi} be the proportion of submarines of type h assigned to the i^{th} mission. Make $X = ||x_{hi}||$ satisfy the conditions

$$\sum_{i} x_{hi} = 1, \text{ for every } h, \; x_{hi} \geq 0,$$

and

$$\alpha_{hi} \leq x_{hi} \leq \beta_{hi}, \text{ for every pair } h,i,$$

where the sets $\{\alpha_{hi}\}$, $\{\beta_{hi}\}$ are supposed to satisfy

$$\sum_{i} \alpha_{hi} < 1 < \sum_{i} \beta_{hi} \text{ for every } h$$

and

$$0 \leq \alpha_{hi} < \beta_{hi} \text{ for every pair } h,i.$$

2. The Space Y

Let y_{jk} be the proportion of antisubmarine forces of type j sent to the k^{th} place. Make $Y = ||y_{jk}||$ satisfy the conditions

$$\sum_k y_{jk} = 1, \text{ for every } j, \; y_{jk} \geq 0$$

and

$$a_{jk} \leq y_{jk} \leq b_{jk} \text{ for every pair } j,k$$

where the sets $\{a_{jk}\}$, $\{b_{jk}\}$ are supposed to satisfy

$$\sum_k a_{jk} < 1, < \sum_k b_{jk}$$

and

$$0 \leq a_{jk} < b_{jk} \text{ for every pair } j,k.$$

3. <u>The Function F(x,y)</u>

V_{him} is the value of accomplishing stage m of mission i for a submarine of type h. The character of F(x,y) can be examined.

Set

$$W_{him} = V_{him} \exp \left[-\theta_{him}\right]$$

where θ_{him} is as before. Put

$$T_{hi} = \sum_m W_{him}.$$

Then

$$F(x,y) = \sum_{h,i}^{m} x_{hi} \, T_{hi}.$$

This function is linear in x and exponential in y and is therefore a concave-convex game of the type treated in [4], defined over X x Y. The quantity F(x,y) represents the total expected payoff to the submarine player. Re-expressing F(x,y) in its explicit form gives:

$$F(x,y) = \sum_h \sum_i x_{hi} \sum_m V_{him} \exp[-\sum_j \sum_k E_{hijkm} C_{hj\ell(k)} y_{jk}].$$

The remainder of this chapter gives details of the application of the algorithm to the game defined above.

The principal result is the flow-chart (Appendix). Since this flow chart is constructed around the algorithm from [4], it is helpful, though not essential, to have [4] available. The complete program listing is included following the Appendix. The program is written in FORTRAN IV and was run on the IBM 360/67 computer at the Naval Postgraduate School, Monterey, California.

B. BASIC COMPUTATIONS

1. Computation of θ_{him}

For fixed (h,i), θ_{him} is non-decreasing in the mission stage m. This reflects the trivial fact that

P[submarine survives stage m]

$\leq$ P[submarine survives stage m-1].

Equality holds when the "threat" due to the encounters at stage m is non-existent, i.e., when either no ASW-forces y_{jk} are present or their effectiveness against the submarine, $C_{hj\ell(k)}$, is zero. Hence θ_{him}, for each pair (h,i), is accumulated over the mission stages as follows:

$$\theta_{him} = \theta_{hi(m-1)} + \sum_j \sum_k A_{hijkm}\, y_{jk}, \quad \theta_{hio} = 0.$$

2. Partial Derivatives with Respect to x_{hi}

Because of the linearity of F in x the partials with respect to x_{hi}, $F_{x_{hi}}$, are the coefficients of x_{hi} and do not explicitly contain x. $V_{him} \exp[-\theta_{him}]$ can be represented as a matrix of dimension (n x m), where n is the number of pairs (h,i). Then

$$F_{x_{hi}} = \sum_m V_{him} \exp[-\theta_{him}]$$

is the sum of the elements in the (h,i) row of that matrix.

3. The Value of F(x,y)

F(x,y) is obtained by pre-multiplying $F_{x_{hi}}$ by x_{hi} and summing the products

$$F(x,y) = \sum_{hi} x_{hi} F_{x_{hi}} .$$

4. Partial Derivatives with Respect to y_{jk}

Because of the cumulative property of θ_{him}, a change in y_{jk} during some mission stage m' will affect the following stages as well. For each pair (h,i), premultiply x_{hi} by the corresponding $A_{hijkm'}$, where m' is the mission stage in which the y_{jk} of interest occurs. Sum $V_{him} \exp[-\theta_{him}]$ over the mission stages for which $m' \leq m$, multiply the result with $x_{hi} A_{hijkm'}$ and sum the products over h and i:

$$-F_{y_{jk}} = \sum_h \sum_i x_{hi} A_{hijkm'} \sum_{m' \leq m} V_{him'} \exp[-\theta_{him'}].$$

5. Finding Directions of Increase (Decrease)

$F_{x_{hi}}$ and $F_{y_{jk}}$ are inputs to the derivative games for x and y. For x, the direction $\gamma°$ is sought such that $D_\gamma \phi(x) = \nabla F_x \cdot \gamma$ is maximized; for y, g° is sought such that $D_g \operatorname{Max}_x F(x,y) =$

$\nabla F_y \cdot g$ is minimized (equivalently, $-\nabla F_y \cdot g$ is maximized). The side condition is that $\gamma°$ and $g°$ be unit vectors:

$$\sum_i \gamma_{hi} = \sum_k g_{jk} = 0, \sum_i \gamma_{hi}^2 = \sum_k g_{jk}^2 = 1; \forall h, \forall j.$$

A method of finding such directions - called THE DIRECTION FINDING ALGORITHM - is derived from the Kuhn-Tucker conditions and the Schwartz Inequality. It is contained in [4] and has been applied to a vector - valued function by Zmuida in [22].

C. PROGRAMMING NOTES

The flow chart (Appendix) and the associated program may not be optimal with respect to machine time and memory space required, and may provide opportunities for improvement. For computer routines like this, intended to solve high-dimensional problems, a trade-off between time and space is generally apparent. The user, considering the particular facilities available to him, must decide on his optimal trade-off, and modify the program accordingly. The following discusses some of the techniques that have been implemented; it points out the major difficulties that have been encountered and the ways presently used to deal with these, and offers some remarks about the impact of the underlying mathematics on the use of the algorithm for realistic problems. The numbers referenced are statement numbers.

1. Presentation of Input - The Mission Description

In a realistic application, most of the A_{hijkm} will be zero because the effectiveness $C_{hj\ell(k)}$ contained in A will be zero. Example: suppose k denotes a place in the western Baltic and $\ell(k)$ classifies this place as shallow with extremely poor sonar conditions; j denotes a nuclear killer-submarine,

h represents a conventional attack submarine. Then $C_{hj\ell(k)}$ will be zero for all practical purposes.

The method of presenting the matrix E which avoids computing of and with A when C is zero, is one which is extremely easy for the user to understand and employ. He gives his description of a submarine mission by plotting it on a chart, marks off in order the places the submarine must go, and notes the forces it might meet at those places. He will give the exposure E required by the particular mission in terms of a standard which he will have set. He will mark off the points in the mission where the various stages of the mission will have been accomplished. Such a description might run as follows:

```
m=0

h=1   i=1   k=1   j=2   E=1.0
                  j=5   E=1.5
            k=7   j=1   E=2.0
                  j=3   E=3.0

m=1 (the first stage of the first mission is completed)

            k=8   j=4   E=1.2
                  j=7   E=1.7

m=2

            k=15  j=3   E=0.3
                  j=2   E=2.0
                  j=4   E=0.5

m=3
```

The above mission (mission 1 for submarines of type 1) has three stages and nine encounters, in four different places. Then the listing starts for the next mission with h=1, i=2, and continues until all missions for all types of submarines have been described in this manner. Note that this listing has already taken into account the classification

of the places k by $\ell(k)$. This is done in such a way that for each encounter on this mission listing the associated $C_{hj\ell(k)}$ is positive. In other words, a place k, and hence an encounter, will appear on the mission listing only if it seems possible that the opponent will allocate forces to that place *and* the corresponding effectiveness against the maximizer's submarine operating in that place is positive.

There are various possible ways of storing the information contained in the mission description in the machine. One way which is very economical in terms of storage space requirements is to read the list encounter by encounter, i.e., card by card, starting with an N=1. Then $A(N) = E \cdot C_{hj\ell(k)}$ and an indicator array

$$P(N) = I \cdot J \cdot K \cdot M \cdot (h-1) + J \cdot K \cdot M \cdot (i-1) + K \cdot M \cdot (j-1)$$
$$+ M \cdot (k-1) + m + 1$$

contain the complete information. I, J, K, M denote the maximum values of the indices i, j, k, m. The reason for using (m+1) is that the correct m appears *after* the mth stage is completed. The disadvantage of this method is that during computations the individual indices must be recomputed from P(N):

$$h = 1 + \left[\frac{P(N)}{I \cdot J \cdot K \cdot M}\right] \qquad [\cdot] \text{ means "the largest integer in } \cdot\text{"}$$

$$\Delta = P(N) - I \cdot J \cdot K \cdot M(h-1)$$

$$i = 1 + \left[\frac{\Delta}{J \cdot K \cdot M}\right]$$

$$\Delta\Delta = \Delta - J \cdot K \cdot M \cdot (i-1)$$

$$j = 1 + \left[\frac{\Delta\Delta}{K \cdot M}\right]$$

$$\Delta\Delta\Delta = \Delta\Delta - K \cdot M(j-1)$$

$$k = 1 + \left[\frac{\Delta\Delta\Delta}{M}\right]$$

$$m = \Delta\Delta\Delta - M(k-1)$$

In the sample program contained in this paper, the indices in their original form are stored in vector arrays and are directly accessible throughout the computations.

2. The Contraction with Respect to V_{him}

This contraction takes into account the possibility that the stage m of a mission may have been noted, but that the associated V_{him} may have been declared to be zero. Since it is useless to calculate anything involving a $V_{him}=0$, the corresponding elements in the index arrays of the mission are eliminated. This is done prior to the execution of the game and may therefore be referred to as the basic contraction.

3. Working in Subspaces

It can be expected that most of the x_{hi} and y_{jk} will be on their boundaries at any stage, including the approximate optimal solution. This suggests the idea of eliminating computations involving variables which have fixed values either temporarily or throughout the process.

To do this, one reduces the dimensions of the spaces, thus saving machine time. One can redefine the space so as to "freeze" the variables on the boundaries leaving the remainder free to move.

Where in the program should redefinition of the Y-space take place? The minimizer can hold the subspace fixed and continue to move in that subspace until an apparent minimum has been reached. At this point the allocation corresponding to this minimum has to be checked for optimality in the full space. This will require at least one iteration through the minimization routine in the full space. After a minimum valid in the full space has been obtained, the Y-space is reduced to a new subspace. An alternative approach is to redefine the space after each change in the Y-allocation, thus maintaining a continously changing subspace. The authors feel that the latter will enable the minimizing variable to stay in the subspace longer finding directions of decrease, before the need arises to employ the full space. This idea was implemented in the program.

The contractions for x (statement 1871) and y (125) eliminate computations involving elements that are on the lower boundaries. It must be noted that the contraction itself and the complications caused by its use take machine time. Worthwhile time savings in computation will not be realized until this technique is applied to relatively large scale problems.

INDX(N), N = 1,...,NNX, contains the indices of the $x_{hi} > 0$, INDY(N), N = 1,...,NNY, contains the indices of the $y_{jk} > 0$. NNX and NNY are the dimensions of the subspaces for X and Y. These index arrays are used to control which

variables are to be involved in the computations. The way this is done in practice can be seen in the program listing.

One remark concerning the DIRECTION FINDING ALGORITHMS, (400) and (1400), may be in order: these algorithms have to be applied row by row to the matrices $||x_{hi}||$ and $||y_{jk}||$. In subspaces, the number of elements belonging to a particular row varies. The outer do-loop runs over the number of rows. To find the numbers of elements per row (these numbers serve as the termination values of the inner do-loops) a test on the indices stored in INDX and INDY is conducted (402 ff), (1402 ff). The test value, NTEST, is the index of the last element in the rows of $||x_{hi}||$ and $||y_{jk}||$, respectively. When the test passes, all elements in the row at hand have been collected and the algorithm begins. Before the next row is picked, NSTART is incremented by the number of elements found in the previous row so that the search through INDX or INDY always starts in the correct position.

4. Machine Accuracy Problems

The program calls for frequent testing of floating point numbers. Throughout the development of the program these tests have been sources of trouble. Testing for equality must be strictly avoided even after - as has been done here in various places - variables close to a fixed quantity have been reset to that quantity (e.g., (1050 ff) or (1060 ff)). Although the program specified double-precision, it took extensive experimentation to maintain

feasibility, i.e., satisfy the side conditions

$$\sum_i x_{hi} = \sum_k y_{jk} = 1$$

to at least the 13th decimal place.

Of critical importance is the accuracy in the computation of the direction matrices γ (GAX) and g (GAY). In the neighborhood of a maximum or minimum, the partials F_x and F_y are close to zero, as are the Lagrange multipliers (XMU and YMU), and the differences F_x-XMU,F_y-YMU (SD). In order to determine the value of the derivative game, the sum of the squares of these differences has to be formed, an operation that may very likely lead to erroneous results which, in turn, carry over into the computation of γ and g.

The countermeasure taken is to premultiply the SD when they are small by a large number, (505 ff), (1510 ff).

5. Testing the Program

As the calling of subroutines is exceedingly time consuming, the present program does not use them. This made debugging tedious. The "standard" routine, i.e., the program employing the derivative games in the original form, was tested running a small scale example where $||x||$ was (2 X 2) and $||y||$ was (2 X 4), making it possible to hand-check the computations.

The "auxiliary game" routine using the B-R process was debugged using the following objective function:

$$F(x,y) = 2x_1 \exp[-2y_1 - y_2] + 2x_2 \exp[-y_1 - 2y_2]$$
$$+ x_3 \exp[-y_1 - y_2 - y_3]$$

Subject to: $\sum_i x_i = \sum_k y_k = 1$

$x_i, y_k \geq 0$

with initial allocations $x = (0,0,1)$, $y = (0,1,0)$. For a discussion of this example in light of the algorithm see [4]. The reason that this can only be solved through the B-R routine lies in the fact that, against the initial x, any y represents an exact minimum, however, the value of the derivative game for x is positive, yielded by $\gamma^\circ = (0, \sqrt{2}/2, -\sqrt{2}/2)$.

Finally, an example with 100 encounters was rigged up where $||x||$ was (4x4), $||y||$ was (5x10). While this does not yet represent a problem in high-dimensional space, the authors are confident that this example provided a sufficiently severe test to demonstrate the validity of the program as written.

6. General Comments

The following remarks are intended to facilitate the use and modification of the program.

a. Initial Allocation

The initial allocations are generated as corner points in the stationary part (99). It can be expected that, in the optimal solution, most of the variables will be on the boundaries. An initial point on the boundaries insures that the number of "absurd" allocations is minimal.

If one were to use an interior point as a starting solution, it possibly would involve large numbers of absurd allocations (e.g., a submarine in an aircraft barrier patrol). The machine would spend an enormous amount of time reducing such allocations to zero.

b. Distance Policy

The initial and re-set values for the distances are determined from the dimensions of the spaces; the user may employ his own rules. A compromise between re-set values too large causing "overshooting" and values to small causing "creeping" should be considered. In general, "creeping" is much costlier in terms of machine time. The policy for halving distances and its rationale is outlined in [4].

c. Upper and Lower Bounds

For simplicity, 0 and 1 have been used in the program. Specifying individual bounds α_{hi}, β_{hi} and a_{jk}, b_{jk} does not introduce additional problems but increases the storage space required considerably (e.g., for y, two additional arrays of the same dimension as y).

d. Modifying the Objective Function

The algorithm as stated is valid for concave-convex functions under quite general conditions. Though the present program has been designed to handle a particular linear-exponential function, it is quite flexible. The objective function mentioned in subsection 5 may serve to illustrate this point:

$$F(x,y) = 2x_1 \exp[-2y_1-y_2] + 2x_2 \exp[-y_1-2y_2] + x_3 \exp[-y_1-y_2-y_3]$$

is produced by a very simple adjustment of the mission description:

```
h=1    i=1    k=1    j=1    E=2.0
              k=2    j=1    E=1.0
              k=3    j=1    E=0.0   END OF MISSION 1
   m=1

h=1    i=2    k=1    j=1    E=1.0
              k=2    j=1    E=2.0
              k=3    j=1    E=0.0   END OF MISSION 2
   m=1

h=1    i=3    k=1    j=1    E=1.0
              k=2    j=2    E=1.0
              k=3    j=3    E=1.0   END OF MISSION 3
   m=1
```

The associated V values are: $V_{111} = 2.0$

$V_{121} = 2.0$

$V_{131} = 1.0$.

The associated $C_{hj\ell(k)}$ are: $C_{111} = C_{112} = C_{113} = 1.0$

e. Assigning Values V_{him}

The values (of completing stage m of mission i for submarine of type h) are relative measures. When assigning V_{him}, the user should consider that a submarine may have spent part of its weapons (resources) during stage m-1. This may reduce its operational capabilities for stage m, and the value for stage m should reflect this fact.

f. Choosing ρ (RHO)

The parameter ρ, mentioned in Chapter II, section B 2.b, has to be chosen by the user. It specifies

the accuracy to which $\operatorname{Max} \phi(x)$ is to approximate the value of the game $F(x,y)$, V. ρ affects the y-minimization by controlling the accuracy, $\varepsilon(d)$, to which the derivative game $D_g \operatorname{Max}_x F(x,y) = -\nabla F_y \cdot g$ has to approximate its mathematical value. The y-minimization is the most time consuming portion of the process.

$$\varepsilon(d) = \frac{\rho \cdot d}{36 \cdot \delta}$$

where δ is the diameter of the X-space. Considering the "worst" case, when $d = d_o = \rho/36 \cdot L$, (L is the maximum oscillation of F_x), it becomes apparent that

$$\varepsilon(d_o) = \frac{\rho^2}{36^2 \cdot \delta \cdot L}$$

is of order of magnitude $\rho^2 \cdot 10^{-3}$.

Recall that the conditions established for the valididty of the algorithm are intended to guarantee that, at the approximate optimal solution, $|\phi(x)-V| \leq \rho$. Test runs of the program seem to indicate that the accuracy actually obtained is much higher.

Although generally valid conclusions cannot be derived from this observation the user must be aware of this feature because he will pay heavily in terms of machine time when ρ is unreasonably small.

IV. SUMMARY

The devlopment of the mathematical theory underlying the derivative game and the concave-convex game algorithm has only been sketched out in this paper. Here, that algorithm has been employed in the formation of a potentially useful example. With programming techniques designed to provide economical running in high dimensions, large-scale problems should be amenable to the application of the concave-convex game algorithm.

V. SUGGESTIONS FOR FURTHER STUDY

A. MATHEMATICS

In Section III.C.6.e., the question of the proper choice of ρ was addressed. Recall that

$$|\max_x \phi(x) - V| \leq \rho.$$

Even when the user has based his choice of a value for ρ on extensive experimentation with a program, he still will not know exactly how close $\max_x \phi(x)$ is to the value of the game. The desirable state of affairs would be

$$|\max_x \phi(x) - V| = \rho$$

This would require the formulation of necessary conditions on the functions α, β and ε.

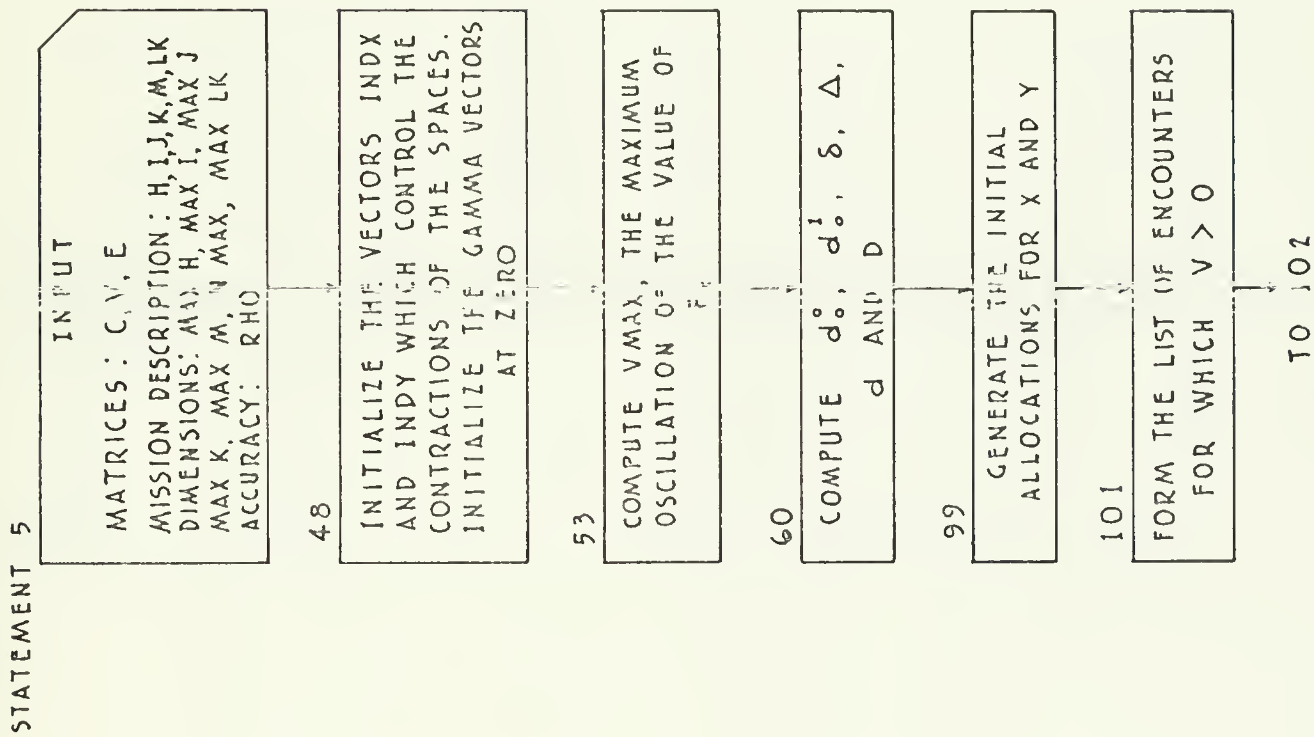
B. PROGRAMMING

The fact that, for the class of games at issue, $\mathrm{Max}_x\mathrm{Min}_y F = \mathrm{Min}_y\mathrm{Max}_x F$ can be exploited to arrive at the neighborhood of the optimal solution faster than the present program will: Start the problem as MinMaxF. The course of action then is reversed with considerable advantages. The maximizer sees the present y-allocation; the maximization is trivial, assigning as much weight as possible to the x_{hi} with the largest coefficients (the $F_x(x,y°)$), which results in a corner solution for x. Then the B-R technique is employed directly to F(x,y) which will bring x off the boundaries

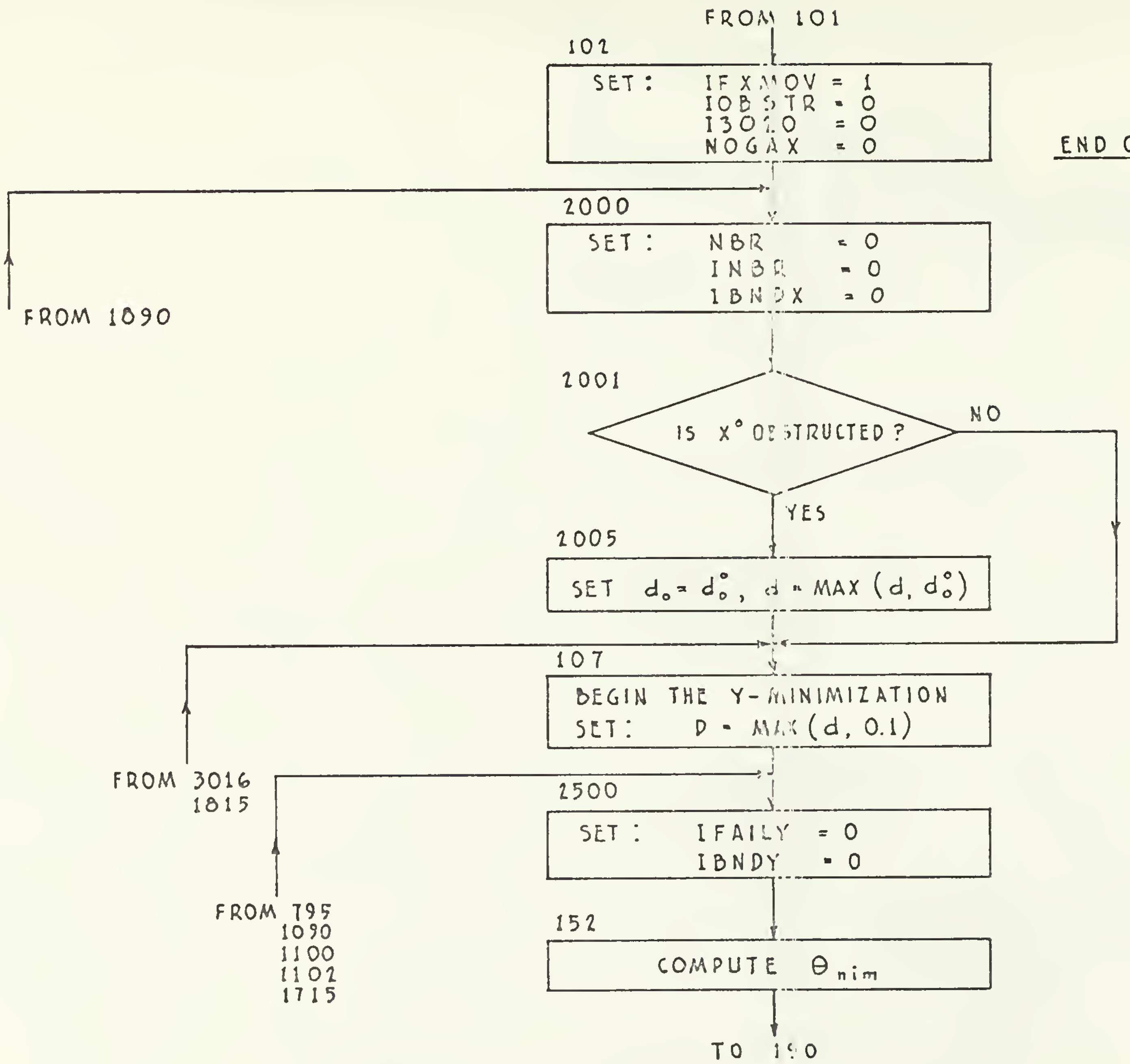
again, and, after a pre-set number of iterations, will produce an allocation not far from the solution. At this stage the problem is reinterpreted as MaxMinF and solved in the manner of the present program.

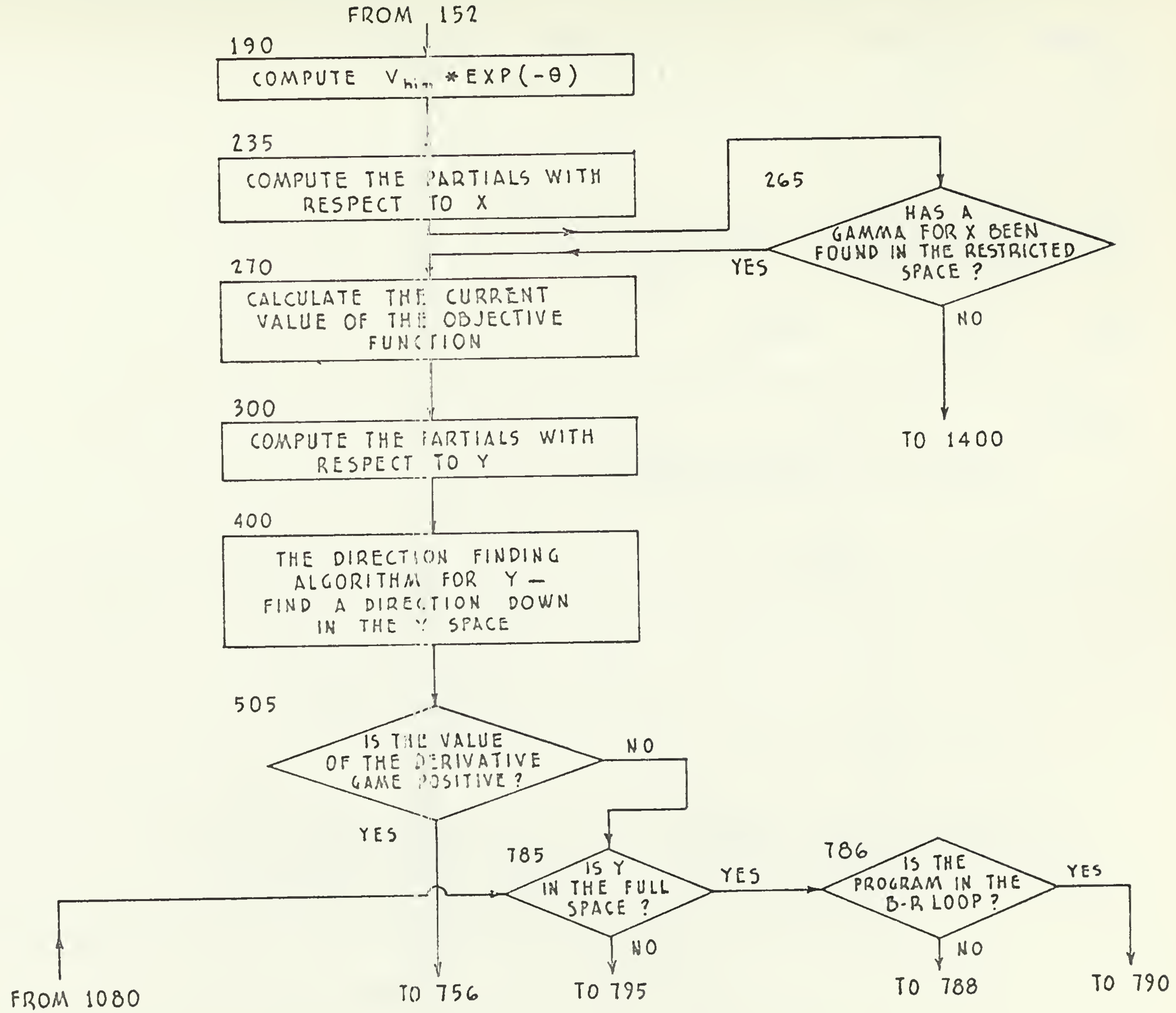
Another feasible refinement particularly useful in the application of the algorithm to objective functions linear in both x and y, i.e., $F = \sum_{ij} x_i a_{ij} y_j$, is the idea of "doubling", outlined in [4]. The problem is treated as MaxMin and MinMax at the same time. Here the value of the game is approached from below and above simultaneously which provides a stopping rule when the difference $\underset{x}{\mathrm{Max}} F(x,y) - \phi(x)$ arrives at a pre-specified value.

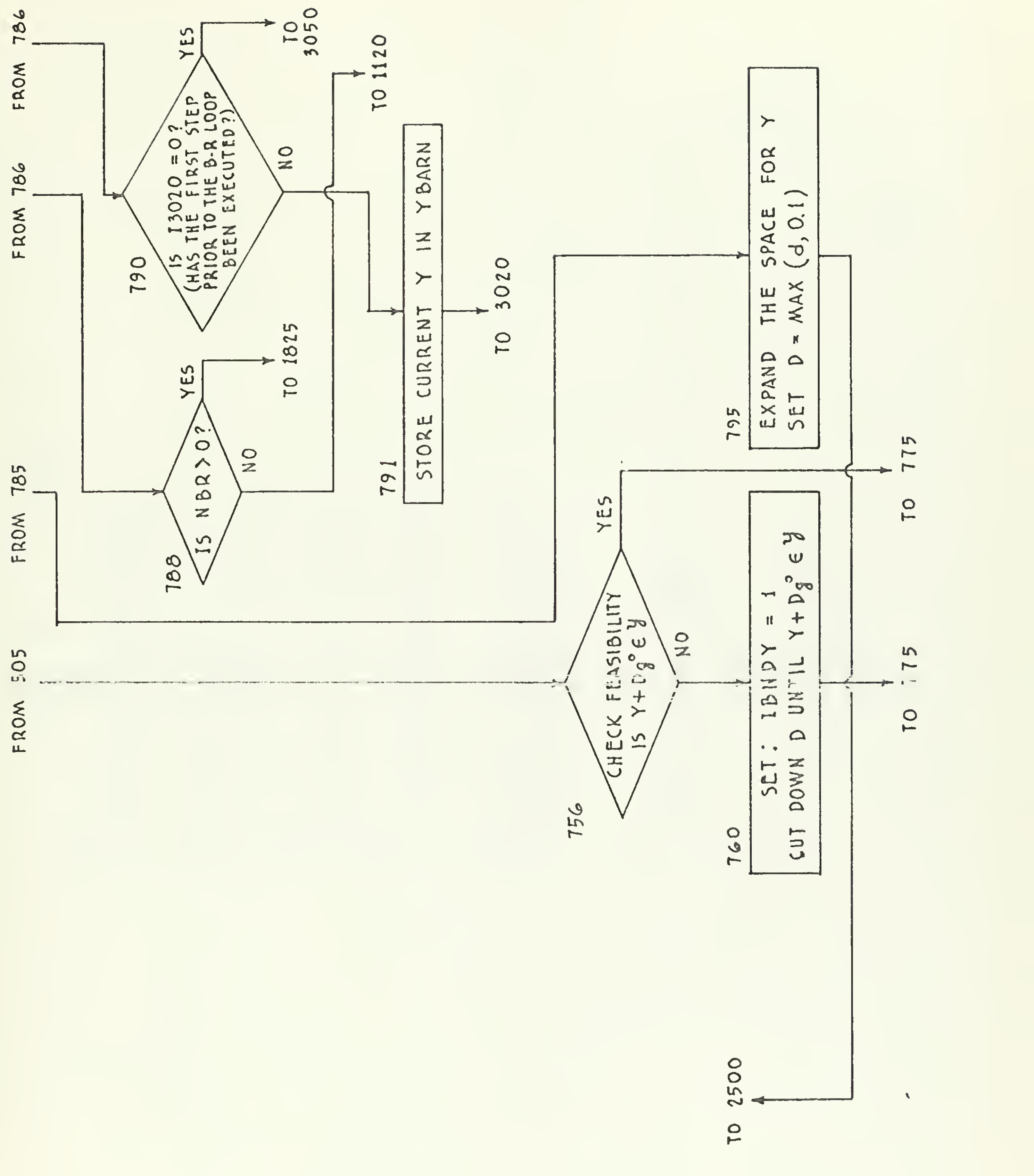
APPENDIX: FLOWCHART OF THE PROGRAMMED EXAMPLE

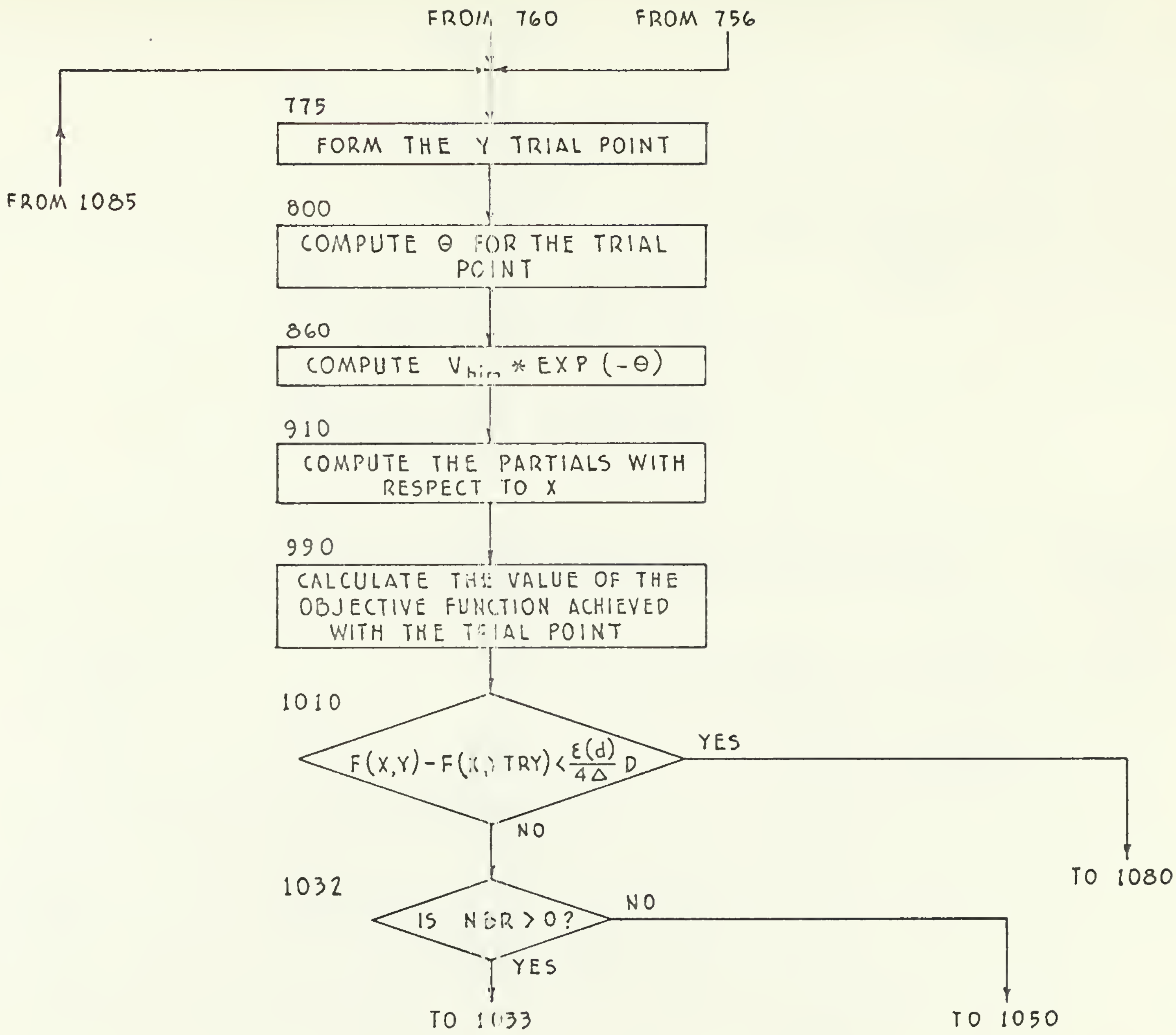


END OF STATIONARY SECTION

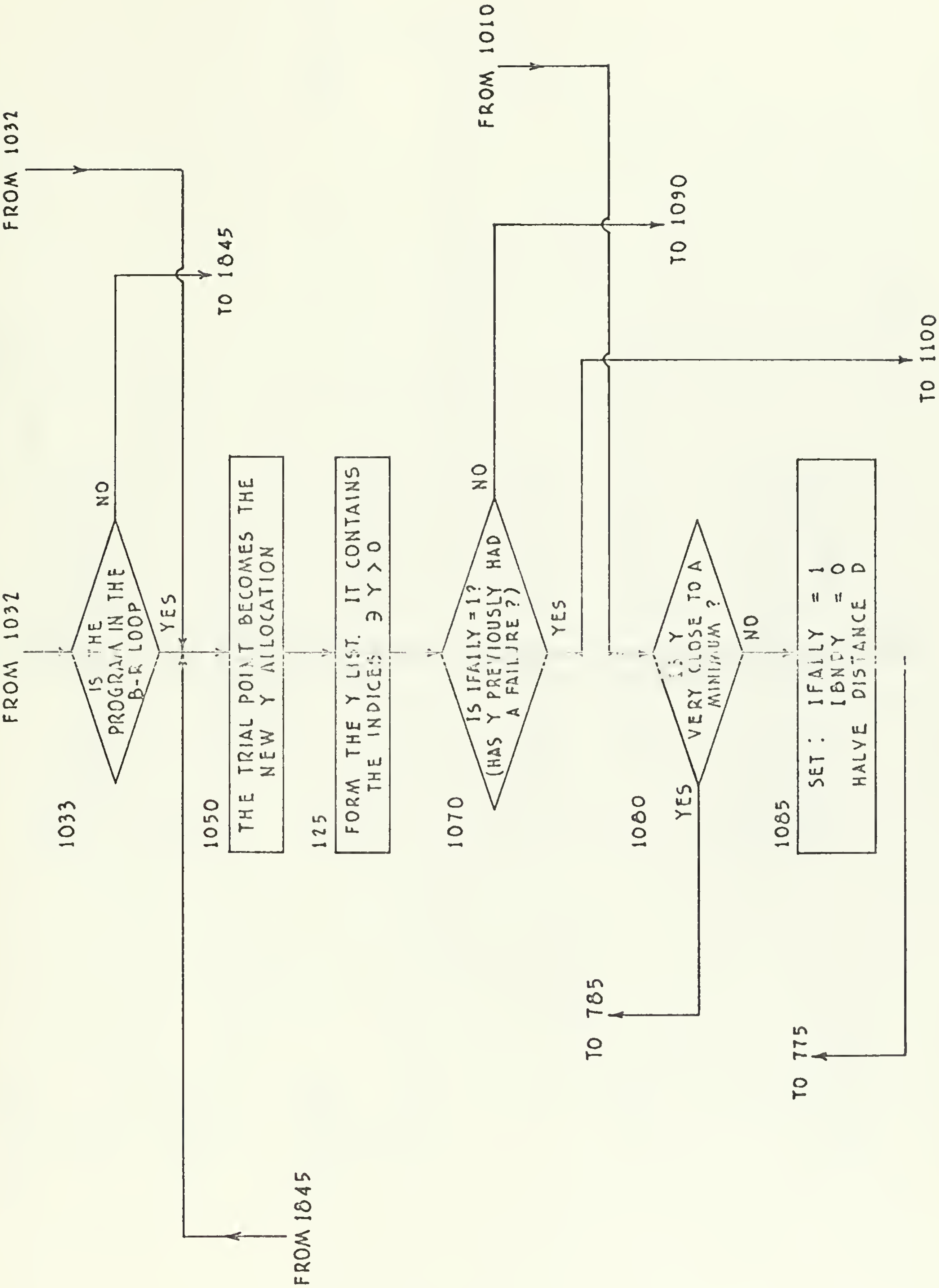


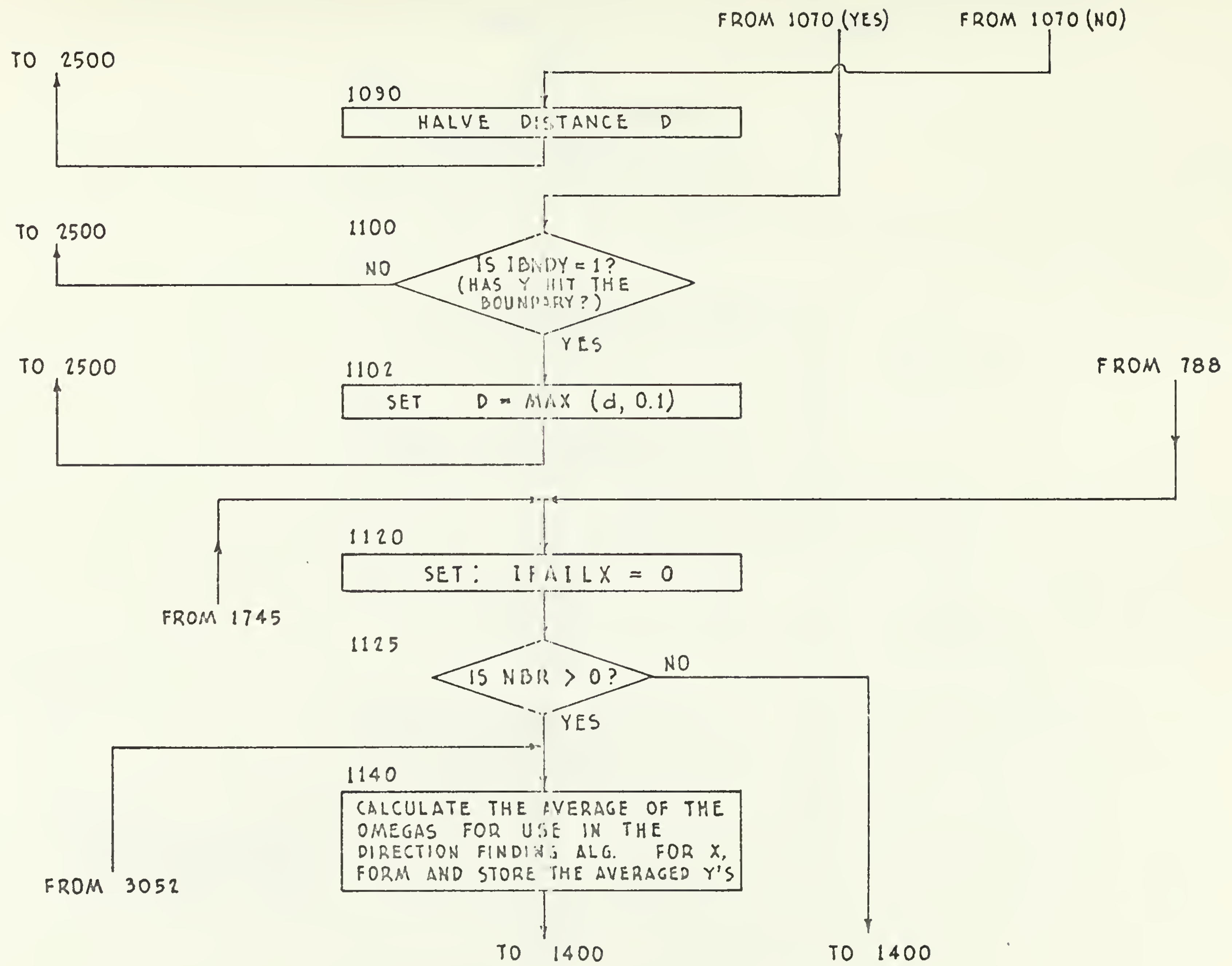

FROM 152
190
COMPUTE V_{him} * EXP(-θ)
235
COMPUTE THE PARTIALS WITH RESPECT TO X
265
HAS A GAMMA FOR X BEEN FOUND IN THE RESTRICTED SPACE ?
YES
NO
TO 1400
270
CALCULATE THE CURRENT VALUE OF THE OBJECTIVE FUNCTION
300
COMPUTE THE PARTIALS WITH RESPECT TO Y
400
THE DIRECTION FINDING ALGORITHM FOR Y — FIND A DIRECTION DOWN IN THE Y SPACE
505
IS THE VALUE OF THE DERIVATIVE GAME POSITIVE ?
NO
YES
785
IS Y IN THE FULL SPACE ?
YES
NO
786
IS THE PROGRAM IN THE B-R LOOP ?
YES
NO
FROM 1080
TO 756
TO 795
TO 788
TO 790

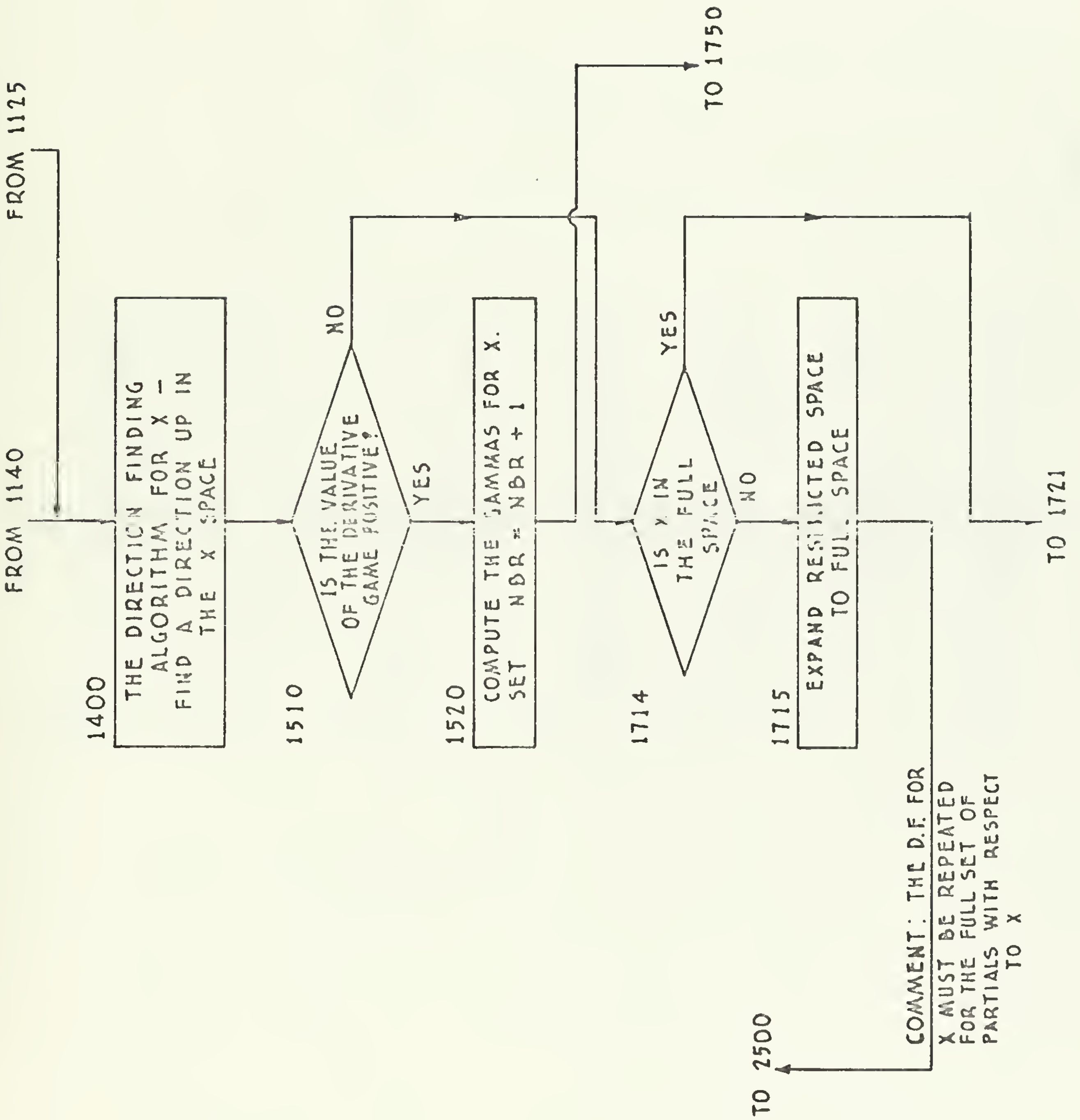

FROM 786
FROM 786
FROM 785
FROM 505
790
IS I3020 = 0?
(HAS THE FIRST STEP PRIOR TO THE B-R LOOP BEEN EXECUTED?)
YES
TO 3050
NO
TO 1120
788
IS NBR > 0?
YES
TO 1825
NO
791
STORE CURRENT Y IN YBARN
TO 3020
795
EXPAND THE SPACE FOR Y
SET D = MAX (d, 0.1)
756
CHECK FEASIBILITY
IS Y + D°g ∈ 𝒴
YES
TO 775
NO
760
SET: IBNDY = 1
CUT DOWN D UNTIL Y + D°g ∈ 𝒴
TO 775
TO 2500

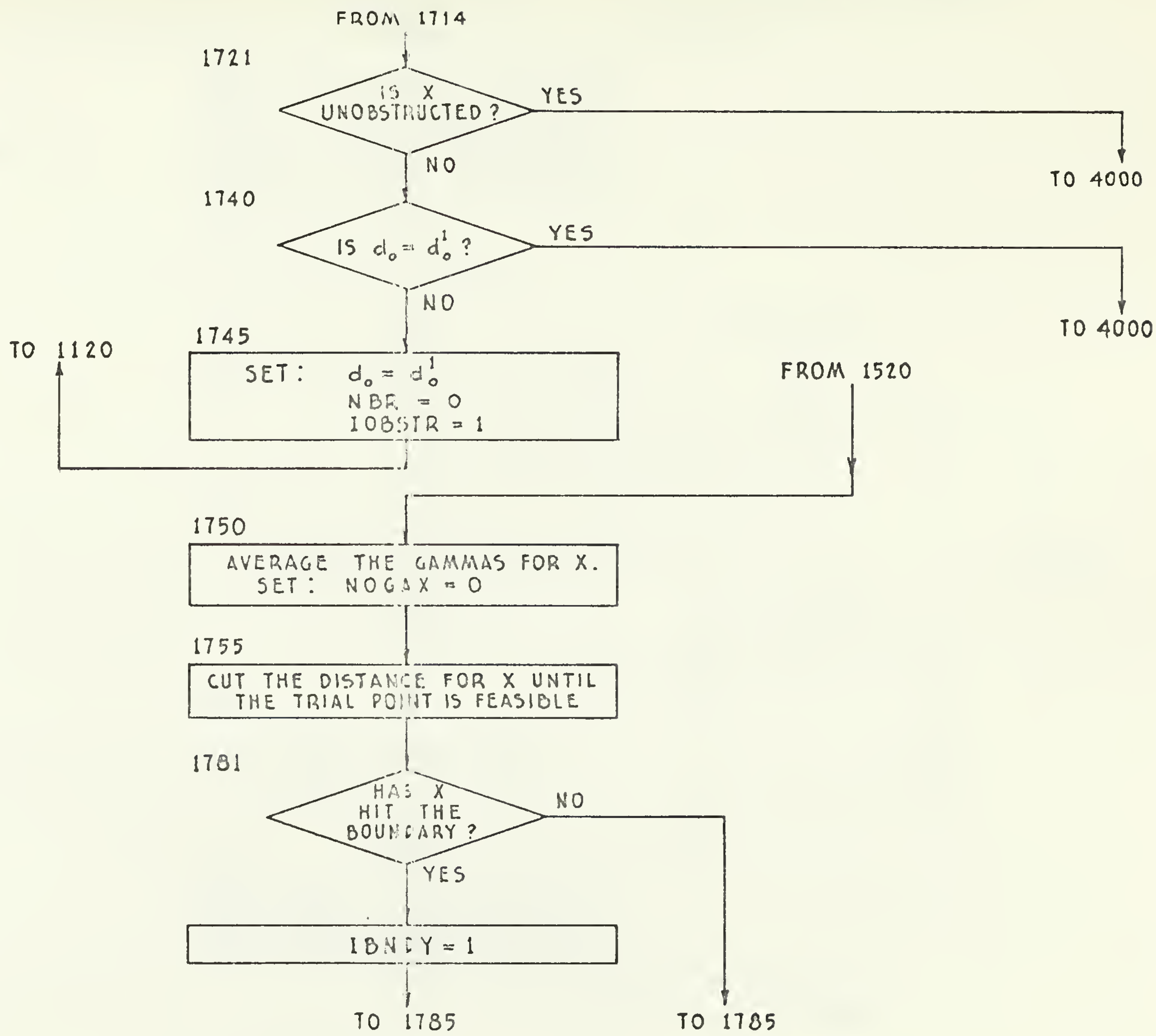


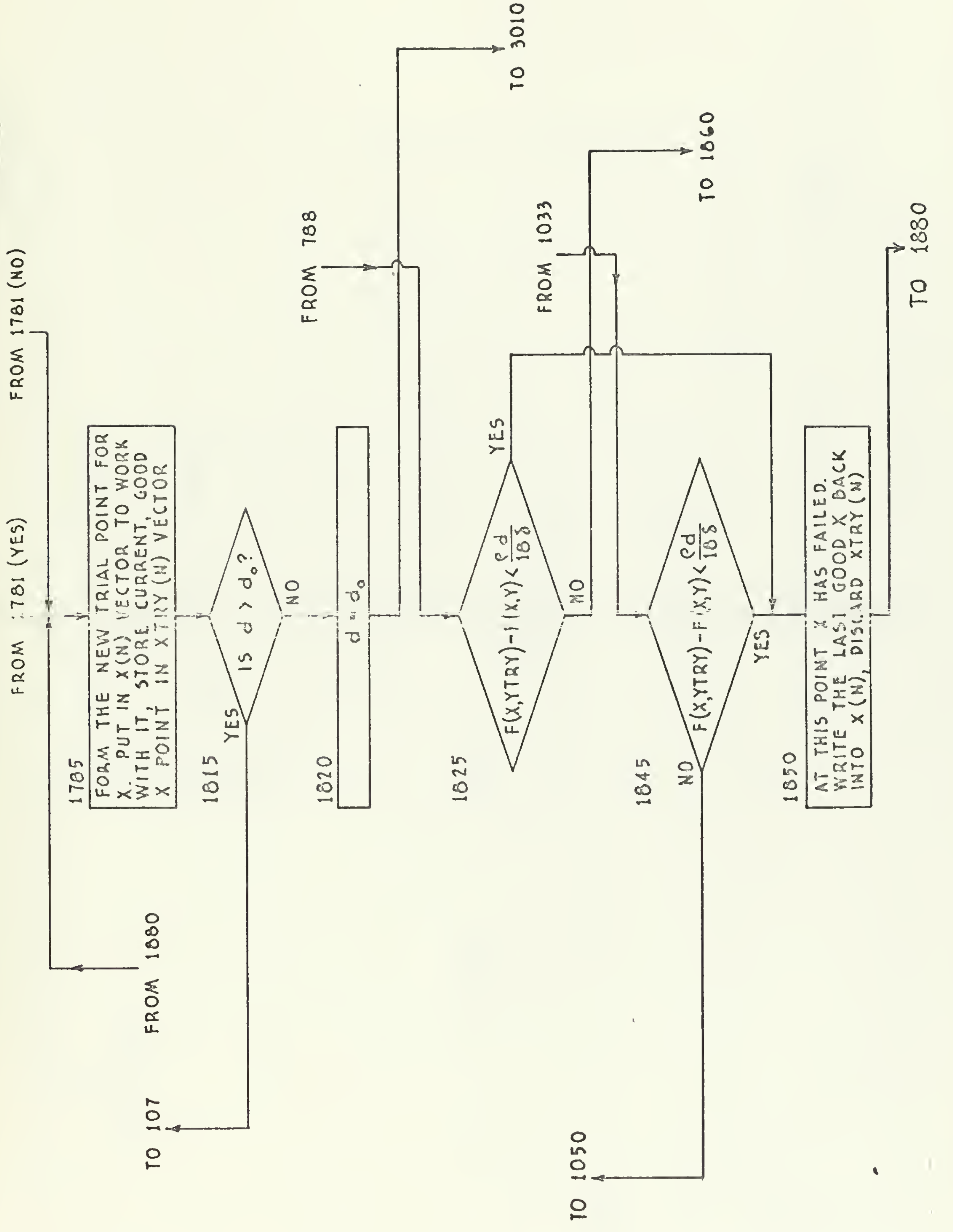
FROM 760
FROM 756
FROM 1085
775
FORM THE Y TRIAL POINT
800
COMPUTE Θ FOR THE TRIAL POINT
860
COMPUTE V * EXP (-Θ)
910
COMPUTE THE PARTIALS WITH RESPECT TO X
990
CALCULATE THE VALUE OF THE OBJECTIVE FUNCTION ACHIEVED WITH THE TRIAL POINT
1010
F(X,Y)-F(X, TRY) < ε(d)/4Δ D
YES
TO 1080
NO
1032
IS NGR > 0?
NO
TO 1050
YES
TO 1033


FROM 1032
1033
IS THE PROGRAM IN THE B-R LOOP
NO
YES
FROM 1032
TO 1845
FROM 1845
1050
THE TRIAL POINT BECOMES THE NEW Y ALLOCATION
125
FORM THE Y LIST. IT CONTAINS THE INDICES ∋ Y > 0
1070
IS IFAILY = 1? (HAS Y PREVIOUSLY HAD A FAILURE ?)
NO
YES
FROM 1010
TO 1090
TO 1100
1080
IS Y VERY CLOSE TO A MINIMUM ?
YES
NO
TO 785
1085
SET : IFAILY = 1
IBNDY = 0
HALVE DISTANCE D
TO 775

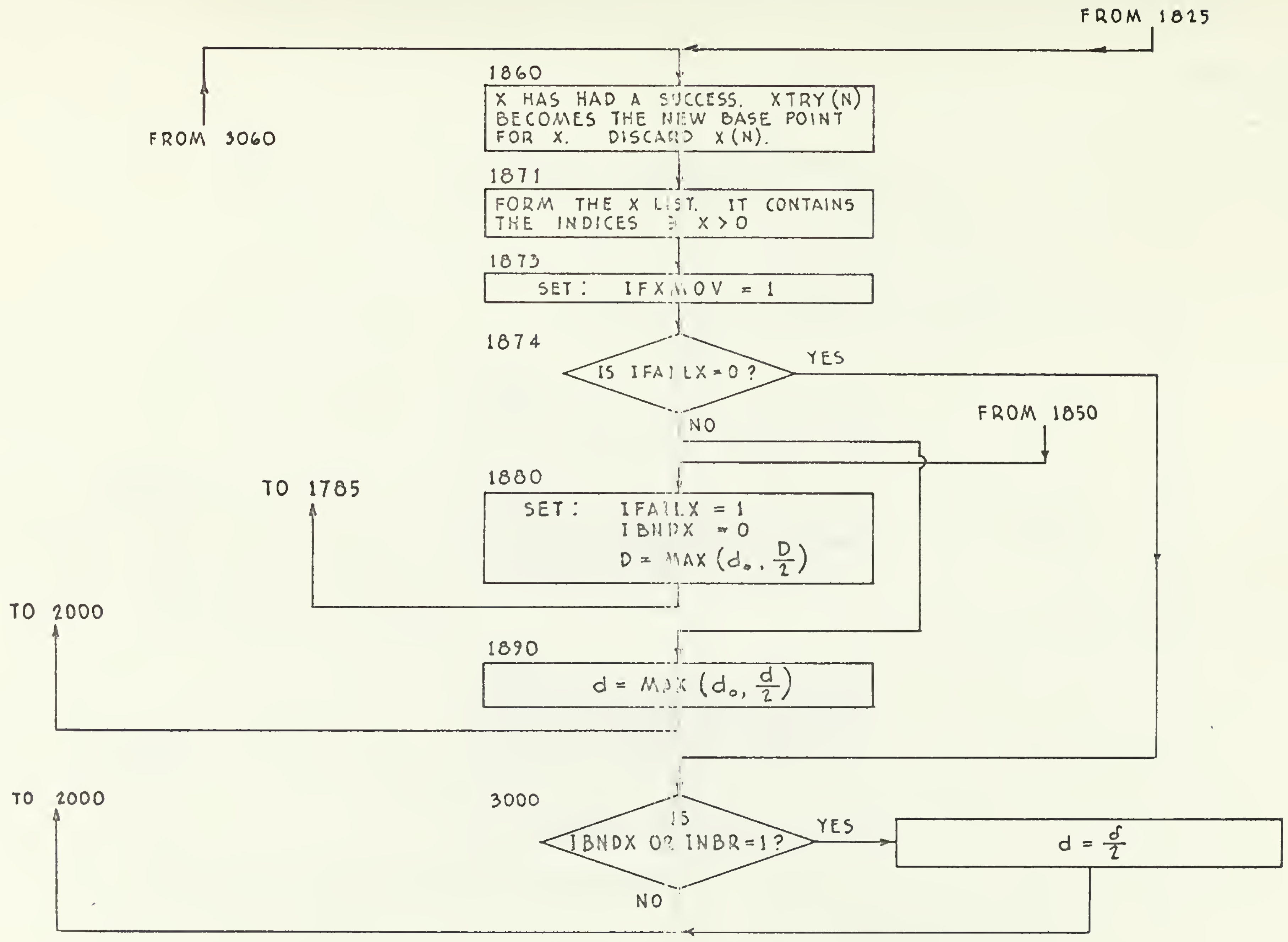

FROM 1070 (YES)
FROM 1070 (NO)
TO 2500
1090
HALVE DISTANCE D
TO 2500
1100
NO
IS IBNDY = 1? (HAS Y HIT THE BOUNDARY?)
YES
TO 2500
1102
SET D = MAX (d, 0.1)
FROM 788
1120
SET: IFAILX = 0
FROM 1745
1125
IS NBR > 0?
NO
YES
1140
CALCULATE THE AVERAGE OF THE OMEGAS FOR USE IN THE DIRECTION FINDING ALG. FOR X, FORM AND STORE THE AVERAGED Y'S
FROM 3052
TO 1400
TO 1400

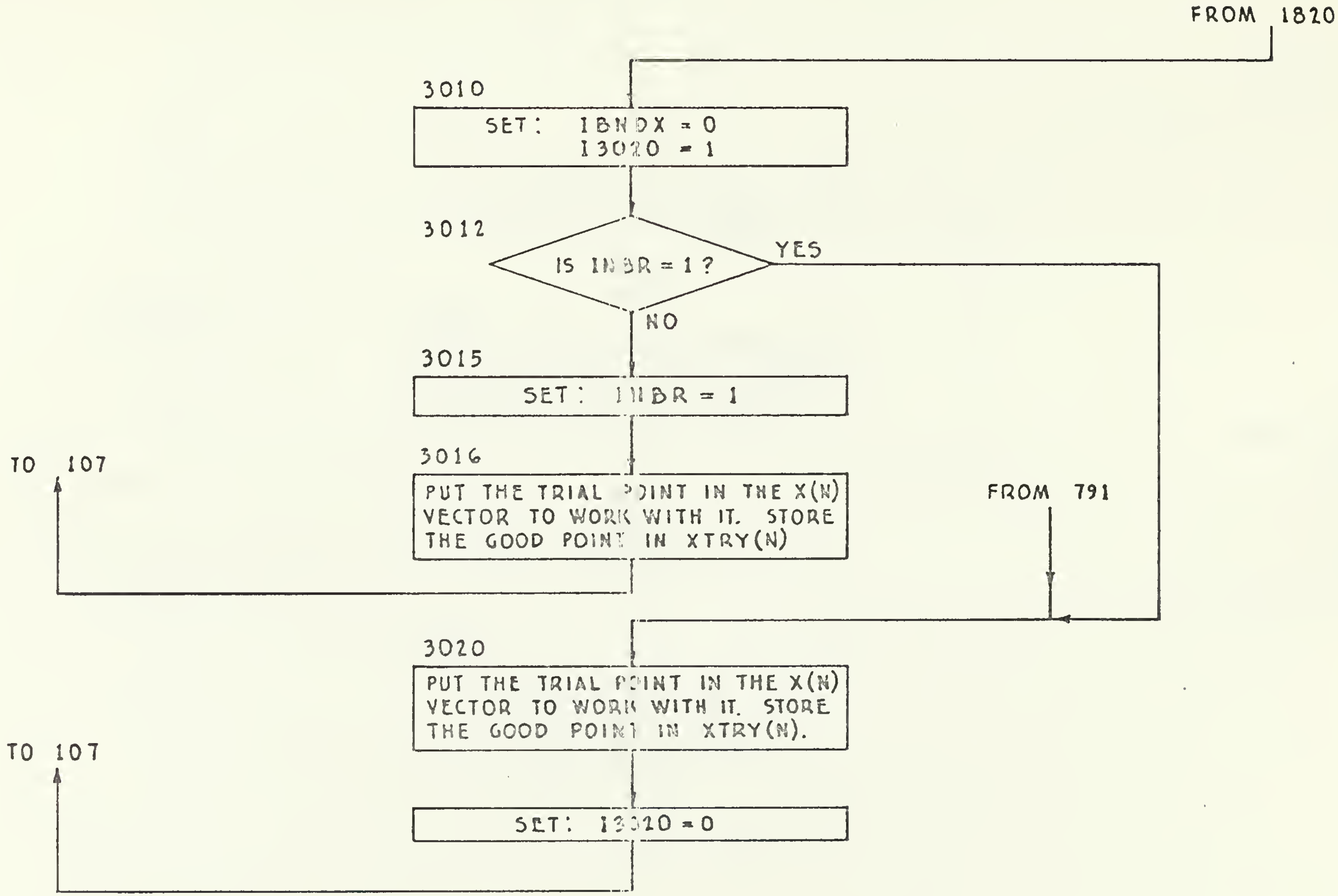

FROM 1125
FROM 1140
1400
THE DIRECTION FINDING ALGORITHM FOR X — FIND A DIRECTION UP IN THE X SPACE
1510
IS THE VALUE OF THE DERIVATIVE GAME POSITIVE?
NO
YES
TO 1750
1520
COMPUTE THE GAMMAS FOR X. SET NBR = NBR + 1
1714
IS X IN THE FULL SPACE
YES
NO
1715
EXPAND RESTRICTED SPACE TO FULL SPACE
TO 1721
COMMENT: THE D.F. FOR X MUST BE REPEATED FOR THE FULL SET OF PARTIALS WITH RESPECT TO X
TO 2500


FROM 1714
1721
IS X UNOBSTRUCTED ?
YES
TO 4000
NO
1740
IS $d_o = d_o^1$?
YES
TO 4000
NO
1745
SET: $d_o = d_o^1$
NBR = 0
IOBSTR = 1
TO 1120
FROM 1520
1750
AVERAGE THE GAMMAS FOR X.
SET: NOGAX = 0
1755
CUT THE DISTANCE FOR X UNTIL THE TRIAL POINT IS FEASIBLE
1781
HAS X HIT THE BOUNDARY ?
NO
TO 1785
YES
IBNDY = 1
TO 1785

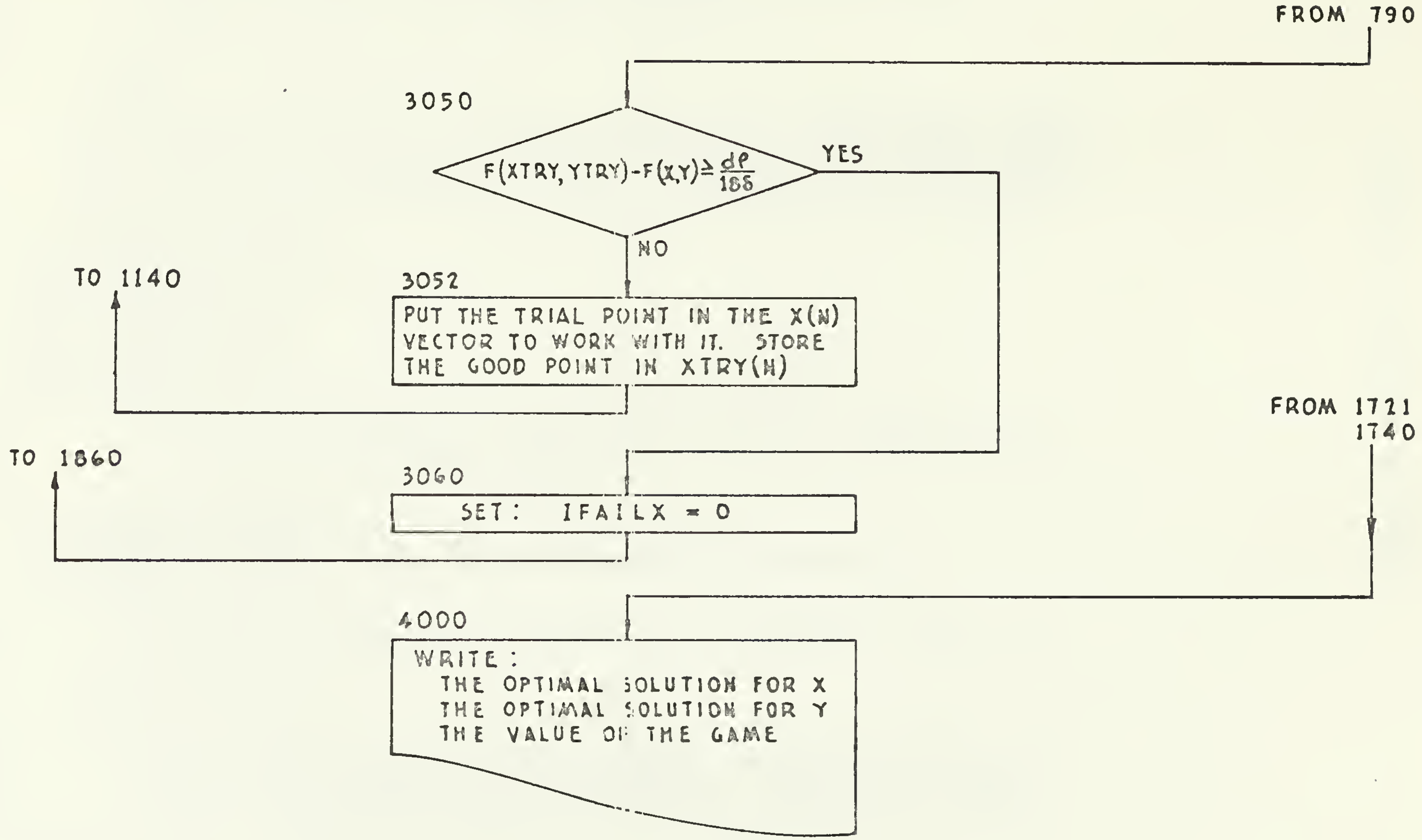


FROM 1781 (YES)
FROM 1781 (NO)
FROM 1880
1785
FORM THE NEW TRIAL POINT FOR X. PUT IN X(N) VECTOR TO WORK WITH IT, STORE CURRENT, GOOD X POINT IN XTRY(N) VECTOR
1815
IS d > d₀?
YES
NO
TO 107
FROM 788
1820
d = d₀
TO 3010
1825
F(X,YTRY)-F(X,Y) < ed/188
YES
NO
FROM 1033
TO 1860
1845
F(X,YTRY)-F(X,Y) < ed/188
YES
NO
TO 1050
1850
AT THIS POINT X HAS FAILED. WRITE THE LAST GOOD X BACK INTO X(N), DISCARD XTRY(N)
TO 1880


FROM 1825
FROM 3060
1860
X HAS HAD A SUCCESS. XTRY(N) BECOMES THE NEW BASE POINT FOR X. DISCARD X(N).
1871
FORM THE X LIST. IT CONTAINS THE INDICES ∋ X > 0
1873
SET: IFXMOV = 1
1874
IS IFAILX = 0 ?
YES
NO
FROM 1850
TO 1785
1880
SET: IFAILX = 1
IBNDX = 0
D = MAX (d₀, D/2)
TO 2000
1890
d = MAX (d₀, d/2)
TO 2000
3000
IS IBNDX OR INBR = 1 ?
YES
NO
d = δ/2



FROM 1820
3010
SET: IBNDX = 0
I3020 = 1
3012
IS INBR = 1 ?
YES
NO
3015
SET: INBR = 1
3016
PUT THE TRIAL POINT IN THE X(N) VECTOR TO WORK WITH IT. STORE THE GOOD POINT IN XTRY(N)
TO 107
FROM 791
3020
PUT THE TRIAL POINT IN THE X(N) VECTOR TO WORK WITH IT. STORE THE GOOD POINT IN XTRY(N).
SET: I3020 = 0
TO 107


FROM 790
3050
F(XTRY,YTRY)-F(X,Y) ≥ dP/188
YES
NO
TO 1140
3052
PUT THE TRIAL POINT IN THE X(N) VECTOR TO WORK WITH IT. STORE THE GOOD POINT IN XTRY(N)
FROM 1721 1740
TO 1860
3060
SET: IFAILX = 0
4000
WRITE:
THE OPTIMAL SOLUTION FOR X
THE OPTIMAL SOLUTION FOR Y
THE VALUE OF THE GAME

COMPUTER PROGRAM LISTING

```
C
C        THIS PROGRAM IS DESIGNED TO DETERMINE THE
C        (APPROXIMATELY) OPTIMAL SOLUTION TO A HIGH-
C        DIMENSIONAL ALLOCATION PROBLEM.  THE  OBJECTIVE
C        FUNCTION IS ** F(X,Y)=SUM OVER H,I OF  X(H,I) *SUM
C        OVER M OF V(H,I,M)*EXP(-THETA(H,I,M) ** WHERE
C        THETA(H,I,M) = SUM OVER J,K OF
C        E(H,I,J,K,M)*C(H,J,L(K))*Y(J,K) ..
C
C
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION X(16),XX(4),INDX(16),GAX(16),GBARX(16),DFX(1
     16),IDSX(4),DDFX(4),XTRY(16),OMBAR(16),Y(50),YY(10),IND
     2Y(50),YBARN(50),GAY(50),DFY(50),IDSY(10),DDFY(10),SD(5
     30),C(80),V(80),NH(100),I(100),J(100),K(100),M(100),LK(
     4100),E(100),A(100),VE(80),THETA(80),YTRY(50),SA(100)
    5 READ(5,10)C,V
   10 FORMAT(16F5.1)
   20 READ(5,25)NH,I,J,M,LK
   25 FORMAT(80I1)
   27 READ(5,28)K
   28 FORMAT(40I2)
   40 READ(5,45)E
   45 FORMAT(16F5.1)
C
C        RHO IS THE ACCURACY SPECIFIED FOR THE COMPUTATION
C        OF THE VALUE OF THE GAME.
C
      READ(5,46)RHO
   46 FORMAT(8F10.6)
      READ(5,47)MAXH,MAXI,MAXJ,MAXK,MAXM,NMAX,MAXLK
   47 FORMAT(16I5)
C
C        MAXX IS THE MAXIMUM NUMBER OF X ELEMENTS, MAXY IS
C        THE SAME FOR Y.
C
      MAXX=MAXH*MAXI
      MAXY=MAXJ*MAXK
C
C        INITIALIZE THE VECTORS INDX AND INDY WHICH CONTROL
C        THE CONTRACTIONS OF THE SPACES.  INITIALIZE THE
C        GAMMA VECTORS AT ZERO.
C
   48 DO 51 N=1,MAXX
      INDX(N)=N
   51 GAX(N)=0.
      NNX=MAXX
      DO 52 N=1,MAXY
```

```
      INDY(N)=N
   52 GAY(N)=0.
      NNY=MAXY
C
C        COMPUTE VMAX, THE MAXIMUM OSCILLATION OF THE VALUE
C        OF DEL DOT THE PARTIAL WITH RESPECT TO X.
C
   53 VMAX=0.
      DO 58 N=1,MAXX
      SUM=0.
      NN=MAXM*(N-1)
      DO 57 L=1,MAXM
      NL=NN+L
   57 SUM=SUM+V(NL)
      IF(SUM.GT.VMAX) VMAX=SUM
   58 CONTINUE
      MAXV=MAXX*MAXM
      MX=MAXI*MAXM
      DMAXH=MAXH
      DMAXJ=MAXJ
C
C        COMPUTE THE VALUES OF THE VARIOUS DISTANCES.
C
C        DOXO AND DOX1 ARE THE PARAMETERS OF ACCURACY
C        FOR THE UNOBSTRUCTED AND OBSTRUCTED X .
C
C        DELTAX AND DELTAY ARE THE DIAMETERS OF THE X AND
C        Y SPACES.
C
   60 DOXO=RHO/(36.*VMAX)
      DOX1=DOXO/(2.*MAXI*DSQRT(DMAXH))
      DELTAX=DSQRT(2.*DMAXH)
      DELTAY=DSQRT(2.*DMAXJ)
      DFORX=0.5*DELTAX
      DFORY=DFORX
      DOX=DOXO
C
C        DFORX AND DFORY ARE THE MAXIMUM DISTANCES THAT
C        X AND Y MAY MOVE AT ANY ONE ITERATION. IN THE
C        PROCESS, DFORY WILL BE RE-SET TO MAX(DFORX,0.1) .
C
C        GENERATE INITIAL ALLOCATIONS FOR X AND Y. THE
C        FIRST ELEMENT IN EACH ROW OF THE X AND Y MATRICES
C        IS SET TO ONE, AND ALL OTHER ELEMENTS TO ZERO.
C
   99 DO 104 N=1,MAXH
      NX=MAXH*(N-1)
      DO 103 L=1,MAXI
```

```
      NI=NX+L
      X(NI)=0.
      IF(L.EQ.1) X(NI)=1.
  103 CONTINUE
  104 CONTINUE
      DO 106 N=1,MAXJ
      NY=MAXK*(N-1)
      DO 105 L=1,MAXK
      NI=NY+L
      Y(NI)=0.
      IF(L.EQ.1) Y(NI)=1.
  105 CONTINUE
  106 CONTINUE
C
C         FORM THE LIST OF ENCOUNTERS FOR WHICH V>0
C
  101 NN=0
   50 DO 100 N=1,NMAX
      NHV=NH(N)
      IV=I(N)
      MV=M(N)
      NHV1=NHV-1
      NV=MX*NHV1+MAXM*(IV-1)+MV
      IF(V(NV).EQ.0.) GO TO 100
   55 NN=NN+1
      NH(NN)=NHV
      I(NN)=IV
      J(NN)=J(N)
      K(NN)=K(N)
      M(NN)=MV
      NC=MAXJ*MAXLK*(NHV-1)+MAXLK*(J(N)-1)+LK(N)
      A(NN)=C(NC)*E(N)
  100 CONTINUE
      NNV=NN
  102 IFXMOV=1
      NOGAX=0
      ICBSTR=0
      I3020=0
 2000 NBR=0
      INBR=0
      IBNDX=0
 2001 IF(IOBSTR.EQ.0)GO TO 107
C
C         NBR IS THE COUNT INDEX FOR THE BROWN-ROBINSON.
C         IBNDX IS 0 IF X HAS NOT HIT THE BOUNDARY AND IS 1
C         IF X HAS HIT THE BOUNDARY.
C
C
```

```
C         THE FOLLOWING TESTS WHETHER X IS OBSTRUCTED AND
C         PICKS  THE DISTANCE TO MOVE ACCORDINGLY.
C
 2005 DOXOC=1.-DOXO
 2010 DO 2020 N=1,NNX
      NX=INDX(N)
      XNX=X(NX)
      IF(XNX.EQ.0.) GO TO 2020
 2012 IF(XNX.EQ.1.) GO TO 2020
 2014 IF(XNX.LT.DOXO) GO TO 107
 2016 IF(XNX.GT.DOXOC) GO TO 107
 2020 CONTINUE
      DOX=DOXO
      D2=DFORX*0.5
      IF(D2.GT.DOXO) GO TO 2024
 2022 DFORX=DOXO
      GO TO 107
 2024 DFORX=D2
C
C         BEGIN THE Y MINIMIZATION.
C
  107 IF(DFORX.GT.0.1) GO TO 140
      DFORY=0.1
      GO TO 2500
  140 DFORY=DFORX
 2500 IFAILY=0
      IBNDY=0
C
C         IFAILY CONTROLS THE HALVING OF THE DISTANCE FOR Y.
C         IT IS 0 IF PREVIOUSLY Y HAD A FAILURE AND 1 IF Y
C         PREVIOUSLY HAD A SUCCESS. IBNDY IS 0 IF Y HAS NOT
C         HIT THE BOUNDARY AND IS 1 IF Y HAS HIT THE BOUNDARY.
C
C         COMPUTE THETA(H,I,M)
C
  152 DO 155 L=1,MAXV
  155 THETA(L)=0.
      NT=NH(1)
      IT=I(1)
      MT=M(1)
  160 DO 180 N=1,NNV
      NN=NH(N)
      II=I(N)
      NX=MAXI*(NN-1)+II
      DO 161 L=1,NNX
      IX=INDX(L)
      IF(NX.EQ.IX) GO TO 162
  161 CONTINUE
```

```
      GO TO 180
  162 JJ=J(N)
      KK=K(N)
      JK=MAXK*(JJ-1)+KK
      MM=M(N)
      IF(NN.GT.NT) GO TO 165
  163 IF(II.GT.IT) GO TO 165
  164 GO TO 168
  165 IF(Y(JK).LE..00000000000001) GO TO 167
  166 MN=MX*(NN-1)+MAXM*(II-1)+MM
      THETA(MN)=A(N)*Y(JK)
  167 NT=NN
      IT=II
      MT=MM
      GO TO 180
  168 IF(MM.GT.MT) GO TO 170
      IF(Y(JK).LE..00000000000001) GO TO 180
  169 MN=MX*(NN-1)+MAXM*(II-1)+MM
      THETA(MN)=THETA(MN)+A(N)*Y(JK)
      GO TO 180
  170 MNN=MX*(NN-1)+MAXM*(II-1)+MT
      MN=MNN-MT+MM
      IF(Y(JK).LE..00000000000001) GO TO 172
  171 THETA(MN)=THETA(MNN)+A(N)*Y(JK)
      GO TO 173
  172 THETA(MN)=THETA(MNN)
  173 MT=MM
  180 CONTINUE
C
C        COMPUTE V(H,I,M)*EXP(-THETA)
C
C        THE INITIAL VALUE FOR V*EXP(-THETA) IS V.
C
  190 DO 200 N=1,MAXV
  200 VE(N)=V(N)
      DO 230 N=1,NNX
      NX=INDX(N)
      NX1=MAXM*(NX-1)
      DO 220 L=1,MAXM
      NV=NX1+L
      IF(V(NV).EQ.0.)GO TO 220
  215 IF(THETA(NV).GT.0.) VE(NV)=VE(NV)*DEXP(-THETA(NV))
  220 CONTINUE
  230 CONTINUE
C
C        V*EXP(-THETA) IS COMPLETE.
C
C
```

```
C         COMPUTE THE PARTIALS WITH RESPECT TO X.  SIMPLY SUM
C         THE ROW OF THE V(H,I,M)*EXP(-THETA) MATRIX.
C
  235 DO 236 N=1,MAXX
  236 DFX(N)=0.
  237 DO 260 N=1,NNX
      NX=INDX(N)
      NX1=MAXM*(NX-1)
      DFDX=0.
  239 DO 240 L=1,MAXM
      NL=NX1+L
      DFDX=DFDX+VE(NL)
  240 CONTINUE
      DFX(NX)=DFDX
  260 CONTINUE
C
C         NOGAX=1 MEANS NO GAMMA FOR X HAS BEEN FOUND IN
C         RESTRICTED SPACE .
C
  265 IF(NOGAX.EQ.1)GO TO 1400
C
C         CALCULATE THE CURRENT VALUE OF THE OBJECTIVE
C         FUNCTION.
C
      FXY=0.
  270 DO 280 N=1,MAXX
      IF(X(N).LE..00000000000001) GO TO 280
  275 FXY=FXY+X(N)*DFX(N)
  280 CONTINUE
      FXYSAV=FXY
      IF(I3020.EQ.1) GO TO 285
      IF(IFXMOV.EQ.0)GO TO 287
  285 FXOYO=FXY
      IFXMOV=0
      GO TO 300
  287 FXTYT=FXY
C
C         COMPUTE THE PARTIALS WITH RESPECT TO Y.
C
C         A  MUST BE RETRIEVED FROM THE LIST.
C
  300 DO 301 N=1,MAXY
  301 DFY(N)=0.
C
  305 DO 330 N=1,NNV
      NN=NH(N)
      II=I(N)
      IX=MAXI*(NN-1)+II
```

```
      IF(X(IX).LE..00000000000001) GO TO 330
  314 JJ=J(N)
      KK=K(N)
      JK=MAXK*(JJ-1)+KK
  316 DO 318 L=1,NNY
      JT=INDY(L)
      IF(JK.EQ.JT) GO TO 320
  318 CONTINUE
      GO TO 330
  320 SA(N)=X(IX)*A(N)
      MM=M(N)
      SUM=0.
  322 DO 325 L=MM,MAXM
      NV=MX*(NN-1)+MAXM*(II-1)+L
      SUM=SUM+VE(NV)
  325 CONTINUE
      DFY(JK)=DFY(JK)+SUM*SA(N)
  330 CONTINUE
C
C        THE DIRECTION FINDING ALGORITHM FOR Y - FIND
C        A DIRECTION DOWN IN THE Y SPACE.
C
  400 NSTART=1
  401 DO 500 NJ=1,MAXJ
      SUM=0.
      NCOUNT=0
      NY=0
C
C        THE LOGIC SWITCH LOGICD IS USED TO DETERMINE
C        WHETHER YMU HAS CHANGED (1) OR NOT (0).
C
      LOGICD=1
C
C        THIS CONTRACTS THE SPACE AND GIVES THE PROPER
C        UPPER BOUND ON DO-LOOP INDICES:  NY= THE NUMBER OF
C        RELEVANT ELEMENTS PER ROW.
C
      NTEST=NJ*MAXK
  402 DO 405 N=NSTART,NNY
      IY=INDY(N)
      IF(IY.GT.NTEST) GO TO 410
  403 NY=NY+1
      DDFY(NY)=DFY(IY)
      YY(NY)=Y(IY)
  405 CONTINUE
  410 DO 420 L=1,NY
      IDSY(L)=0
      IF(YY(L).LE..00000000000001) GO TO 420
```

```
  411 IF(YY(L).GE..99999999999999) GO TO 420
  412 IDSY(L)=1
      NCOUNT=NCOUNT+1
      SUM=SUM+DDFY(L)
  420 CONTINUE
      IF(NCOUNT.EQ.0) GO TO 430
  425 YMU=SUM/NCOUNT
      SMALL=YMU
      GO TO 450
  430 SMALL=DDFY(1)
  435 DO 440 L=1,NY
      ZJK=DDFY(L)
      IF(ZJK.LT.SMALL) SMALL=ZJK
  440 CONTINUE
      YMU=SMALL
C
C        NOW GO TO THE UPPER BOUND.
C
  450 MARK=0
  460 DO 470 L=1,NY
      IF(IDSY(L).EQ.1)GO TO 470
  461 IF(YY(L).LT..99999999999999) GO TO 470
  462 IF(DDFY(L).GE.SMALL)GO TO 470
  463 SMALL=DDFY(L)
      MARK=L
  470 CONTINUE
C
C        TEST WHETHER SMALL IS SMALLER THAN YMU.
C
      IF(SMALL.GE.YMU) GO TO 490
  472 LOGICD=1
      YMU=(NCOUNT*YMU+SMALL)/(NCOUNT+1)
      NCOUNT=NCOUNT+1
      SMALL=YMU
      IDSY(MARK)=1
      GO TO 460
C
C        NOW AS MANY ELEMENTS AS POSSIBLE HAVE BEEN
C        DISTINGUISHED ON THE UPPER BOUND.  GO TO THE
C        LOWER BOUND.
C
  475 MARK=0
      DO 480 L=1,NY
      IF(IDSY(L).EQ.1)GO TO 480
  476 IF(YY(L).GT..00000000000001) GO TO 480
  477 IF(DDFY(L).LE.SMALL) GO TO 480
  478 SMALL=DDFY(L)
      MARK=L
```

```
  480 CONTINUE
      IF(SMALL.LE.YMU) GO TO 495
  481 LOGICD=1
C
C        RECALCULATE YMU.
C
      YMU=(NCOUNT*YMU+SMALL)/(NCOUNT+1)
      NCOUNT=NCOUNT+1
      IDSY(MARK)=1
      SMALL=YMU
      GO TO 475
  490 IF(LOGICD.EQ.0) GO TO 496
C
C        THIS MEANS THAT NO MORE ELEMENTS CAN BE
C        DISTINGUISHED.
C
      GO TO 475
  495 LOGICD=0
      GO TO 450
  496 DO 499 L=1,NY
      NL=NSTART+L-1
      IF(IDSY(L).EQ.1)GO TO 498
  497 SD(NL)=0.
      GO TO 499
  498 SD(NL)=DDFY(L)-YMU
  499 CONTINUE
      NSTART=NSTART+NY
  500 CONTINUE
C
C        FORM THE TOTAL SUM OVER THE ENTIRE DISTINGUISHED
C        SET.
C
C        IS THE VALUE OF THE DERIVATIVE GAME POSITIVE?
C
  505 TOTAL=0.
  510 DO 515 N=1,NNY
  515 TOTAL=TOTAL+SD(N)**2
C
C        IF THIS TEST FAILS, PREMULTIPLY SD BEFORE
C        SQUARING TO PREVENT THE MACHINE FROM SETTING THEM
C        EQUAL TO ZERO.  STEP UP THE TEST VALUE
C        PROPORTIONATELY.
C
      IF(TOTAL.GT..00000001) GO TO 518
      TOTAL=0.
      DO 517 N=1,NNY
      SD(N)=SD(N)*100000000.
  517 TOTAL=TOTAL+SD(N)**2
```

```
      EPSLON=(RHO*DFORX*100000000.)/(36.*DELTAX*DELTAY)
      EPSLON=EPSLON**2
      IF(TOTAL.LE.EPSLON) GO TO 785
      GO TO 520
  518 EPSLON=(RHO*DFORX)/(36.*DELTAX*DELTAY)
      EPSLON=EPSLON**2
      IF(TOTAL.LE.EPSLON) GO TO 785
  520 YLA=DSQRT(TOTAL)
      DO 525 N=1,NNY
  525 GAY(N)=SD(N)/YLA
C
C        AFTER COMPUTING THE TEST VALUE OF Y IN THE PRO-
C        POSED DIRECTION OF MOVE, DETERMINE WHETHER Y WOULD
C        LEAVE ITS SPACE AND CUT DOWN THE  DISTANCE AS
C        NECESSARY. THEN RECOMPUTE F(X,Y) WITH THE NEW
C        TRIAL VALUE FOR Y. THE TEST ON F WILL BE PASSED
C        AFTER FINITELY MANY TRIALS.
C
  756 DTEST=DFORY
  760 DO 770 N=1,NNY
      IF(GAY(N).EQ.0.) GO TO 770
      JY=INDY(N)
      YTEST=Y(JY)+DFORY*GAY(N)
      IF(YTEST.LT.0.) DFORY=-Y(JY)/GAY(N)
      IF(YTEST.GT.1.) DFORY=(1.-Y(JY))/GAY(N)
  770 CONTINUE
C
C        IF Y HAS HIT A BOUNDARY, SET IBNDY = 1 .
C
      IF(DTEST.GT.DFORY) IBNDY=1
C
C        FORM THE Y TRIAL POINT.
C
  775 DO 776 L=1,MAXY
  776 YTRY(L)=0.
  777 DO 780 N=1,NNY
      JY=INDY(N)
      YTRY(JY)=Y(JY)+DFORY*GAY(N)
      IF(YTRY(JY).GT..0000000000001)GO TO 779
      YTRY(JY)=0.
      GO TO 780
  779 IF(YTRY(JY).LT..9999999999999)GO TO 780
      YTRY(JY)=1.
  780 CONTINUE
      GO TO 800
C
C
C
```

```
C
C         THIS BLOCK CONTROLS THE FOLLOWING:
C         IF THE Y-MIN HAS BEEN OBTAINED IN THE RESTRICTED
C         SPACE, THE MINIMUM IS CHECKED FOR VALIDITY IN THE
C         FULL SPACE;  AFTER X HAS MOVED, Y MINIMIZES
C         COMPLETELY;  WHEN IN B-R, Y MINIMIZES COMPLETELY;
C         WHEN IN B-R X USES THE WHOLE SPACE.
C
C
C         IS Y IN THE FULL SPACE?
C
  785 IF(NNY.LT.MAXY)GO TO 795
C
C         IS THE PROGRAM IN THE B-R LOOP?
C
  786 IF(INBR.EQ.0)GO TO 788
  787 GO TO 790
  788 IF(NBR.EQ.0)GO TO 1120
  789 GO TO 1825
C
C         HAS THE FIRST STEP PRIOR TO THE B-R LOOP BEEN
C         EXECUTED?
C
  790 IF(I3020.EQ.0) GO TO 3050
      DO 791 N=1,MAXY
  791 YBARN(N)=Y(N)
      GO TO 3020
C
C         EXPAND THE SPACE FOR Y.  SET DFORY=MAX(DFORX,0.1).
C         STORE CURRENT Y IN YBARN.
C
  795 DO 796 N=1,MAXY
  796 INDY(N)=N
      NNY=MAXY
      IF(DFORX.GT.0.1) GO TO 798
      DFORY=0.1
      GO TO 2500
  798 DFORY=DFORX
      GO TO 2500
C
C         NOW, USING THIS TEST VALUE FOR Y, THE OBJECTIVE
C         FUNCTION CAN BE EVALUATED.  A COMPARISON WILL THEN
C         BE MADE TO DETERMINE WHETHER Y HAS SUCCESSFULLY
C         DEMONSTRATED THE ABILITY TO COUNTER THE CURRENT X
C
C
C         COMPUTE THETA FOR THE TRIAL POINT.
C
  800 DO 801 N=1,MAXV
```

```
801 THETA(N)=0.
    NT=NH(1)
    IT=I(1)
    MT=M(1)
802 DO 850 N=1,NNV
    NN=NH(N)
    II=I(N)
    NX=MAXI*(NN-1)+II
803 DO 805 LX=1,NNX
    IX=INDX(LX)
    IF(NX.EQ.IX) GO TO 807
805 CONTINUE
    GO TO 850
807 JJ=J(N)
    KK=K(N)
    JK=MAXK*(JJ-1)+KK
    MM=M(N)
    IF(NN.GT.NT)GO TO 810
808 IF(II.GT.IT)GO TO 810
809 GO TO 815
810 IF(YTRY(JK).EQ.0.) GO TO 812
811 MN=MX*(NN-1)+MAXM*(II-1)+MM
    THETA(MN)=A(N)*YTRY(JK)
812 NT=NN
    IT=II
    MT=MM
    GO TO 850
815 IF(MM.GT.MT)GO TO 820
816 IF(YTRY(JK).EQ.0.) GO TO 850
817 MN=MX*(NN-1)+MAXM*(II-1)+MM
    THETA(MN)=THETA(MN)+A(N)*YTRY(JK)
    GO TO 850
820 MNN=MX*(NN-1)+MAXM*(II-1)+MT
    MN=MNN-MT+MM
    IF(YTRY(JK).EQ.0.) GO TO 824
822 THETA(MN)=THETA(MNN)+A(N)*YTRY(JK)
    GO TO 826
824 THETA(MN)=THETA(MNN)
826 MT=MM
850 CONTINUE
C
C      COMPUTE V(H,I,M)*EXP(-THETA).
C
860 DO 870 N=1,MAXV
870 VE(N)=V(N)
875 DO 900 N=1,NNX
    NX=INDX(N)
    NX1=MAXM*(NX-1)
```

```
  880 DC 895 L=1,MAXM
      NV=NX1+L
      IF(V(NV).EQ.0.)GO TO 895
  885 IF(THETA(NV).GT.0.) VE(NV)=VE(NV)*DEXP(-THETA(NV))
  895 CONTINUE
  900 CONTINUE
C
C         V*EXP(-THETA) IS COMPLETE.
C
C         COMPUTE THE PARTIALS WITH RESPECT TO X.
C         SIMPLY SUM THE ROW OF THE V*EXP(-THETA)-MATRIX
C
  910 DO 920 N=1,MAXX
  920 DFX(N)=0.
  930 DO 980 N=1,NNX
      NX=INDX(N)
      NX1=MAXM*(NX-1)
      DFDX=0.
  940 DO 950 L=1,MAXM
      NL=NX1+L
      DFDX=DFDX+VE(NL)
  950 CONTINUE
      DFX(NX)=DFDX
  980 CCNTINUE
C
C         CALCULATE THE VALUE OF THE OBJECTIVE FUNCTION
C         ACHIEVED WITH THE TRIAL PCINT.
C
  990 FXY=0.
  995 DO 1000 N=1,MAXX
      IF(X(N).LE..00000000000001) GO TO 1000
  997 FXY=FXY+X(N)*DFX(N)
 1000 CCNTINUE
      FXYT=FXY
C
C         TEST THE NEWLY CALCULATED VALUE OF THE CBJECTIVE
C         FUNCTION.
C
 1010 FDIFF=FXYSAV-FXYT
      EPSLON=(RHO*DFORX*DFORY)/(144.*DELTAX*DELTAY)
 1020 IF(FDIFF.LT.EPSLON) GO TO 1080
 1032 IF(NBR.EQ.0) GO TO 1050
 1033 IF(INBR.EQ.1)GO TO 1050
 1035 GO TO 1845
C
C         THE TRIAL POINT BECOMES THE NEW Y ALLOCATION.
C
 1050 DO 1060 N=1,MAXY
```

```
      Y(N)=YTRY(N)
      YBARN(N)=Y(N)
C
C        THE FOLLOWING TESTING BLOCK IS NECESSARY TO ENSURE
C        THAT THE TESTS FOR Y ON THE BOUNDARIES WILL BE
C        PROPERLY HANDLED BY THE MACHINE.
C
      IF(Y(N).GT..0000000000001) GO TO 1053
 1051 Y(N)=0.
      GO TO 1060
 1053 IF(Y(N).LT..9999999999999) GO TO 1060
      Y(N)=1.
 1060 CONTINUE
C
C        FORM THE Y LIST.  IT CONTAINS THE INDICES SUCH THAT
C        Y>0.
C
  125 NY=0
  129 DO 150 N=1,MAXY
      IF(Y(N).GT.0.) GO TO 145
      GO TO 150
  145 NY=NY+1
      INDY(NY)=N
  150 CONTINUE
      NNY=NY
C
 1070 IF(IFAILY.EQ.0)GO TO 1100
 1071 GO TO 1090
C
C        IS Y VERY CLOSE TO THE MINIMUM?
C
 1080 EPSLON=(RHO*DFORX)/(36.*DELTAX)
      YLAD=YLA*DFORY
      IF(YLAD.LE.EPSLON) GO TO 785
 1085 IFAILY=1
      IBNDY=0
      DFORY=0.5*DFORY
      GO TO 775
 1090 DFORY=0.5*DFORY
      GO TO 2500
 1100 IF(IBNDY.EQ.1) GO TO 1102
 1101 GO TO 2500
 1102 IF(DFORX.GT.0.1) GO TO 1104
 1103 DFORY=0.1
      GO TO 2500
 1104 DFORY=DFORX
 1105 GO TO 2500
C
```

```
C         THE MAXIMIZATION OF X BEGINS.  SINCE THE CURRENT
C         VALUES OF THE PARTIALS WITH RESPECT TO X HAVE BEEN
C         OBTAINED PRIOR TO THIS BUT IN RELATION TO THE
C         CORRECT ALLOCATIONS, THESE PARTIALS SERVE AS THE
C         INPUT TO THE DIRECTION FINDING ALGORITHM FOR X.
C
 1120 IFAILX=0
 1122 DO 1123 N=1,MAXX
 1123 OMBAR(N)=DFX(N)
C
C         IT IS NECESSARY NOW TO DETERMINE WHETHER OR NOT THE
C         BROWN-ROBINSON ROUTINE IS IN USE.  IF IT IS, THE
C         AVERAGE OF THE OMEGAS MUST BE CALCULATED PRIOR TO
C         ENTERING THE DF ALGORITHM FOR X.
C         FORM AND STORE THE AVERAGED Y'S.
C
 1125 IF(NBR.EQ.0)GO TO 1400
C
C         CALCULATE THE AVERAGE OF THE OMEGAS FOR USE IN THE
C         DIRECTION FINDING ALGORITHM FOR X.  FORM AND STORE
C         THE AVERAGED Y'S.
C
 1140 DO 1145 N=1,MAXY
      YBARN(N)=(NBR*YBARN(N)+Y(N))/(NBR+1)
 1145 CONTINUE
      DO 1147 N=1,MAXX
      OMBAR(N)=((NBR-1)*OMBAR(N)+DFX(N))/NBR
 1147 DFX(N)=OMBAR(N)
C
C         THE DIRECTION FINDING ALGORITHM FOR X - FIND A
C         DIRECTION UP IN THE X SPACE.
C
 1400 NSTART=1
      DOXC=1.-DOX
 1401 DO 1500 NJ=1,MAXH
      SUM=0.
      NCOUNT=0
      NX=0
C
C         THE LOGIC SWITCH LOGICD IS USED TO DETERMINE
C         WHETHER XMU HAS CHANGED (1) OR NOT (0).
C
      LOGICD=1
C
C         THIS CONTRACTS THE SPACE AND GIVES THE PROPER
C         UPPER BOUND ON DO-LOOP INDICES:  NX=NUMBER OF
C         RELEVANT ELEMENTS PER ROW.
C
```

```
      NTEST=NJ*MAXI
 1402 DO 1405 N=NSTART,NNX
      IX=INDX(N)
      IF(IX.GT.NTEST) GO TO 1410
 1403 NX=NX+1
      DDFX(NX)=DFX(IX)
      XX(NX)=X(IX)
 1405 CCNTINUE
 1410 DO 1420 L=1,NX
      IDSX(L)=0
C
C        IN CHECKING THE BCUNDARIES, ALLOWANCE IS MADE FOR
C        THE PCSSIBILITY OF X BEING OBSTRUCTED.
C
      IF(XX(L).LT.DOX) GO TO 1420
 1412 IF(XX(L).GT.DOXC) GO TO 1420
      IDSX(L)=1
      NCOUNT=NCOUNT+1
      SUM=SUM+DDFX(L)
 1420 CCNTINUE
      IF(NCOUNT.EQ.0)GO TO 1430
 1425 XMU=SUM/NCOUNT
      SMALL=XMU
      GO TO 1450
 1430 SMALL=DDFX(1)
 1435 DO 1440 L=1,NX
      OMEGA=DDFX(L)
      IF(CMEGA.LT.SMALL)SMALL=OMEGA
 1440 CONTINUE
      XMU=SMALL
C
C        NCW GO TC THE UPPER BOUND.
C
 1450 MARK=0
 1460 DO 1470 L=1,NX
      IF(IDSX(L).EQ.1) GO TO 1470
 1461 IF(XX(L).LT.DOXC) GO TO 1470
 1462 IF(DDFX(L).GE.SMALL)GO TO 1470
 1463 SMALL=DDFX(L)
      MARK=L
 1470 CONTINUE
C
C        TEST WHETHER SMALL IS SMALLER THAN XMU.
C
      IF(SMALL.GE.XMU)GO TO 1490
 1472 LOGICD=1
      XMU=(NCOUNT*XMU+SMALL)/(NCOUNT+1)
      NCOUNT=NCOUNT+1
```

See note on inside rear cover

```
C          TEST THE VALUE OF THE DERIVATIVE GAME.  IS IT
C          POSITIVE?
C
      TOTAL=0.
 1510 DO 1515 N=1,NNX
 1515 TOTAL=TOTAL+SD(N)**2
C
C          IF THIS TEST FAILS, PREMULTIPLY SD BEFORE
C          SQUARING TO PREVENT THE MACHINE FROM SETTING THEM
C          EQUAL TO ZERO.  STEP UP THE TEST VALUE
C          PROPORTIONATELY.
C
      IF(TOTAL.GT..00000001) GO TO 1518
      TOTAL=0.
      DO 1517 N=1,NNX
      SD(N)=SD(N)*100000000.
 1517 TOTAL=TOTAL+SD(N)**2
      EPSLON=(13.*RHO*100000000.)/(36.*DELTAX)
      EPSLON=EPSLON**2
      IF(TOTAL.LE.EPSLON) GO TO 1714
      GO TO 1520
 1518 EPSLON=(13.*RHO)/(36.*DELTAX)
      EPSLON=EPSLON**2
      IF(TOTAL.LE.EPSLON) GO TO 1714
C
C          COMPUTE THE GAMMAS FOR X.
C
 1520 XLA=DSQRT(TOTAL)
      DO 1522 N=1,MAXX
 1522 GAX(N)=0.
      DO 1525 N=1,NNX
      IX=INDX(N)
 1525 GAX(IX)=SD(N)/XLA
 1705 NBR=NBR+1
      GO TO 1750
 1714 IF(NNX.EQ.MAXX) GO TO 1720
C
C          EXPAND THE RESTRICTED X SPACE TO FULL SPACE.
C
 1715 DO 1716 N=1,MAXX
 1716 INDX(N)=N
      NNX=MAXX
C
C          THE DIRECTION FINDING ALGORITHM FOR X MUST BE
C          REPEATED WITH THE FULL SET OF PARTIALS WITH
C          RESPECT TO X.
C
      NCGAX=1
```

```
      GC TO 2500
 1720 DOXC=1.-DOX
      DO 1730 N=1,MAXX
      XN=X(N)
C
C         IS X UNOBSTRUCTED?
C
 1721 IF(XN.EQ.0.) GO TO 1730
 1722 IF(XN.EQ.1.) GO TO 1730
 1724 IF(XN.LT.DCX) GO TO 1740
 1726 IF(XN.GT.DOXC) GO TO 1740
 1730 CCNTINUE
      GO TO 4000
C
C         IF THERE WAS NO DIRECTION UP AND X WAS FREE TO MOVE
C         (UNOBSTRUCTED) ,OR THE DISTANCE TO MOVE WAS ALREADY
C         AT THE MINIMUM, THEN THE PROBLEM IS SOLVED .
C         4000 WRITES SOLUTION AND THEN STOPS PROGRAM.
C
 1740 IF(DOX.EQ.DCX1) GO TO 4000
 1745 DCX=DOX1
      NBR=0
      ICBSTR=1
      GC TO 1120
C
C         AVERAGE THE GAMMAS FOR X.
C
 1750 DO 1752 N=1,MAXX
 1752 GBARX(N)=((NBR-1)*GBARX(N)+GAX(N))/NBR
      NOGAX=0
C
C         CUT THE DISTANCE UNTIL THE TRIAL POINT X IS FEASIBLE.
C
 1755 DTEST=DFORX
 1760 DO 1780 N=1,NNX
      IX=INDX(N)
      IF(GBARX(IX).EQ.0.) GO TO 1780
      XTEST=X(IX)+DFORX*GBARX(IX)
      IF(XTEST.LT.0.) DFORX=-X(IX)/GBARX(IX)
      IF(XTEST.GT.1.) DFORX=(1.-X(IX))/GBARX(IX)
 1780 CONTINUE
C
C         HAS X HIT THE BOUNDARY?
C
 1781 IF(DTEST.GT.DFORX) IBNDX=1
C
C
C
```

```
C         FORM THE NEW TRIAL POINT FOR X.  PUT IN X(N) VECTOR
C         TO WORK WITH IT, STORE CURRENT, GOOD X POINT IN
C         XTRY(N) VECTOR.
C
 1785 DO 1786 N=1,MAXX
 1786 XTRY(N)=0.
 1787 DO 1790 N=1,NNX
      IX=INDX(N)
 1790 XTRY(IX)=X(IX)+DFORX*GBARX(IX)
 1800 DO 1810 N=1,MAXX
      XHOLD=XTRY(N)
      XTRY(N)=X(N)
      X(N)=XHOLD
 1810 CONTINUE
C
C         THIS TEST DIRECTS THE PROGRAM INTO THE B-R ROUTINE
C         BY ITS FAILURE.
C
 1815 IF(DFORX.GT.DOX)GO TO 107
 1820 DFORX=DOX
      GO TO 3010
 1825 FDIFF=FXYT-FXOYO
      EPSLON=(RHO*DFORX)/(18.*DELTAX)
 1830 IF(FDIFF.LT.EPSLON) GO TO 1850
 1831 GO TO 1860
C
 1845 FDIFF=FXYT-FXOYO
      EPSLON=(RHO*DFORX)/(18.*DELTAX)
      IF(FDIFF.LT.EPSLON) GO TO 1850
 1847 GO TO 1050
C
C         AT THIS POINT X HAS FAILED.  WRITE THE LAST GOOD
C         X BACK INTO X(N), DISCARD XTRY(N).
C
 1850 DO 1852 N=1,MAXX
 1852 X(N)=XTRY(N)
      GO TO 1880
C
C         X HAS HAD A SUCCESS.  XTRY(N) BECOMES THE NEW BASE
C         POINT FOR X.  DISCARD X(N).
C
 1860 DO 1870 N=1,NNX
      NX=INDX(N)
      IF(X(NX).GE..0000000000001)GO TO 1863
      X(NX)=0.
      GO TO 1870
 1863 IF(X(NX).LE..9999999999999)GO TO 1870
      X(NX)=1.
```

```
 1870 CONTINUE
C
C         WHEN NOT IN B-R ROUTINE REFORM THE X-LIST. IT
C         CONTAINS THE INDICES SUCH THAT X>0 .
C
      IF(INBR.EQ.1) GO TO 1873
 1871 NX=0
      DO 1872 N=1,MAXX
      IF(X(N).LE..0000000000001)GO TO 1872
      NX=NX+1
      INDX(NX)=N
 1872 CONTINUE
      NNX=NX
 1873 IFXMOV=1
 1874 IF(IFAILX.EQ.0)GO TO 3000
 1875 GO TO 1890
 1880 IFAILX=1
      IBNDX=0
      DTEST=DFORX*0.5
      IF(DTEST.GT.DOX) GO TO 1885
 1882 DFORX=DOX
      GO TO 1785
 1885 DFORX=DTEST
      GO TO 1785
 1890 DTEST=DFORX*0.5
      IF(DTEST.GT.DOX) GO TO 1895
 1892 DFORX=DOX
      GO TO 2000
 1895 DFORX=DTEST
      GO TO 2000
 3000 IF(IBNDX.EQ.1) DFORX=DELTAX*0.5
      IF(INBR.EQ.1) DFORX=DELTAX*0.5
      GO TO 2000
 3010 IBNDX=0
      I3020=1
 3012 IF(INBR.EQ.1)GO TO 3020
 3015 INBR=1
C
C         PUT THE TRIAL POINT IN THE X(N) VECTOR TO WORK
C         WITH IT.  STORE THE GOOD POINT IN XTRY(N).
C
 3016 DO 3017 N=1,MAXX
      XHOLD=X(N)
      X(N)=XTRY(N)
 3017 XTRY(N)=XHOLD
      GO TO 107
C
C
```

```
C        PUT THE TRIAL POINT IN THE X(N) VECTOR TO WORK
C        WITH IT.  STORE THE GOOD POINT IN XTRY(N).
C
 3020 DO 3021 N=1,MAXX
      XHOLD=X(N)
      X(N)=XTRY(N)
      XTRY(N)=XHOLD
 3021 CONTINUE
      I3020=0
      GO TO 107
 3050 FDIFF=FXTYT-FXOYO
      EPSLON=(DOX*RHO)/(18.*DELTAX)
      IF(FDIFF.GE.EPSLON) GO TO 3060
C
C        PUT THE TRIAL POINT IN THE X(N) VECTOR TO WORK
C        WITH IT.  STORE THE GOOD POINT IN XTRY(N).
C
 3052 DO 3053 N=1,MAXX
      XHOLD=X(N)
      X(N)=XTRY(N)
      XTRY(N)=XHOLD
 3053 CONTINUE
      GO TO 1140
 3060 IFAILX=0
      GO TO 1860
C
C        WRITE THE (APPROXIMATELY) OPTIMAL STRATEGIES FOR X
C        AND FOR Y AND THE VALUE OF THE GAME.
C
 4000 WRITE(6,4001)
 4001 FORMAT('1',30X,'THE OPTIMAL SOLUTION FOR X')
      WRITE(6,4002)(X(N),N=1,4)
      WRITE(6,4002)(X(N),N=5,8)
      WRITE(6,4002)(X(N),N=9,12)
      WRITE(6,4002)(X(N),N=13,16)
 4002 FORMAT('0',30X,4F16.10)
      WRITE(6,4003)
 4003 FORMAT('0',30X,'THE OPTIMAL SOLUTION FOR Y')
C
C        IN CASE THE PROGRAM ARRIVES AT A SOLUTION WHILE
C        IN THE B-R ROUTINE, THE AVERAGED Y'S REPRESENT THE
C        OPTIMAL STRATEGY FOR Y.
C
      IF(INBR.EQ.0)GO TO 4008
      DO 4007 N=1,MAXY
 4007 Y(N)=YBARN(N)
 4008 WRITE(6,4004)(Y(N),N=1,10)
      WRITE(6,4004)(Y(N),N=11,20)
```

```
      WRITE(6,4004)(Y(N),N=21,30)
      WRITE(6,4004)(Y(N),N=31,40)
      WRITE(6,4004)(Y(N),N=41,50)
 4004 FORMAT('0',10X,10F10.6)
      WRITE(6,4005)
 4005 FORMAT('0',30X,'THE VALUE OF THE GAME')
      WRITE(6,4006)FXCYO
 4006 FORMAT('0',45X,F12.6)
 9999 STOP
      END
```

BIBLIOGRAPHY

1. Danskin, J. M., "Fictitious Play for Continuous Games", Naval Research Logistics Quarterly, 1, 1955, pp. 313-320.

2. Danskin, J. M., "The Theory of Max-Min, With Applications", Journal of the Society of Industrial and Applied Mathematics, v. 14, 1966, pp. 641-664.

3. Danskin, J. M., The Theory of Max-Min, Springer-Verlag, New York, Inc., 1967.

4. Danskin, J. M., "The Derivative Game in Max-Min and the Solution of Concave-Convex Games", to appear Journal of the Society of Industrial and Applied Mathematics.

5. Dresher, M., Games of Strategy, Theory and Applications, Prentice-Hall, Englewood Cliffs, New Jersey, 1961.

6. Isaacs, R., Differential Games, John Wiley and Sons, Inc., New York, 1965.

7. Kakutani, S., "A Generalization of Brouwer's Fixed Point Theorem", Duke Mathematical Journal, v. 8, No. 3, 1941, pp. 457-459.

8. Karlin, S., Mathematical Methods and Theory in Games, Programming and Economics, Vols. I and II, Addison-Wesley, Reading, Massachusetts, 1959.

9. Koopman, B. O., Search and Screening, Operations Evaluation Group Report No. 56, Washington, D. C., 1946.

10. Kuhn, H. W., and Tucker, A. W., (eds.), Contributions to the Theory of Games, Vol. I, Annals of Mathematics Study No. 24, Princeton University Press, Princeton, New Jersey, 1950.

11. Morse, P. M., and Kimball, G. W., Methods of Operations Research, John Wiley and Sons, Inc., New York, 1951.

12. Robinson, J., "An Iterative Method of Solving a Game", Annals of Mathematics, v. 54, 1951, pp. 296-301.

13. Schroeder, R. G., Mathematical Models of Antisubmarine Tactics, Ph.D. Thesis, Northwestern University, Evanston, Illinois, 1966.

14. Schroeder, R. G., Search and Evasion Games, Naval Postgraduate School Technical Report/Research Paper No. 68, Naval Postgraduate School, Monterey, California, 1966.

15. Suppes, P., and Atkinson, R. C., Markov Learning Models for Multiperson Interactions, Stanford University Press, Stanford, California, 1960.

16. Taylor, J. G., Application of Differential Games to Problems of Military Conflict: Tactical Allocation Problems - Part I, Naval Postgraduate School, Monterey, California, 1970.

17. Taylor, J. G., Application of Differential Games to Problems of Naval Warfare: Surveillance - Evasion - Part I, Naval Postgraduate School, Monterey, California, 1970.

18. von Neumann, J., "Zur Theorie der Gesellschaftsspiele", Mathematische Annalen, Vol. 100, 1928, pp. 295-320.

19. Valentine, F., Convex Sets , McGraw-Hill, New York, 1964.

20. Weyl, H., "Elementary Proof of a Minimax Theorem Due to von Neumann", Contributions to the Theory of Games, Vol. I, Annals of Mathematics Study No. 24, Princeton University Press, Princeton, New Jersey, 1950, pp. 19-25.

21. Williams, J. D., The Compleat Strategyst, McGraw-Hill, New York, 1954.

22. Zmuida, P. T., An Algorithm for Optimization of Certain Allocation Models, M.S. Thesis, Naval Postgraduate School, Monterey, California, 1971.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	Chief of Naval Personnel Pers-11b Department of the Navy Washington, D. C. 20370	1
4.	Department of Operations Research and Administrative Sciences Naval Postgraduate School Monterey, California 93940	1
5.	Professor W. P. Cunningham, Code 55Cm Department of Operations Research and Administrative Sciences Naval Postgraduate School Monterey, California 93940	1
6.	Professor J. M. Danskin, Code 55Dk Department of Operations Research and Administrative Sciences Naval Postgraduate School Monterey, California 93940	2
7.	LT Lee Fredric Gunn, USN 10002 Enford Court Fairfax, Virginia 22030	1
8.	Kapitänleutant Peter Tocha, German Navy c/o Alo Terhoeven 415 Krefeld Marienstr. 43 Germany	1

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)* Naval Postgraduate School Monterey, California 93940	2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED 2b. GROUP
3. REPORT TITLE AN ALGORITHM FOR THE SOLUTION OF CONCAVE-CONVEX GAMES	
4. DESCRIPTIVE NOTES *(Type of report and, inclusive dates)* Master's Thesis; September 1971	
5. AUTHOR(S) *(First name, middle initial, last name)* Lee F. Gunn Peter Tocha	

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
September 1971	78	22

8a. CONTRACT OR GRANT NO. b. PROJECT NO c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) 9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*
10. DISTRIBUTION STATEMENT Approved for public release; distribution unlimited.	
11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY Naval Postgraduate School Monterey, California 93940

13. ABSTRACT

A master's thesis which discusses the solution of concave-convex games. An algorithm is developed, a computer program written and applied to an anti-submarine warfare force allocation problem as an illustration. Techniques for handling concave-convex problems in high dimensions are included.

UNCLASSIFIED
Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Concave-Convex Games						
Games						
ASW Forces						
ASW Force Allocations						
Game Theory						
Resource Allocations						
Allocations						
Allocation Problems						
Concave-Convex Functions						
Derivative Games						
Brown-Robinson Process						
MaxMin						
Directional Derivative						
Algorithms						

DD FORM 1 NOV 65 1473 (BACK)
S/N 0101-807-6821

UNCLASSIFIED
Security Classification

A-31409

Page 67 not included in any copies